Retroforum

4/6

Microcomputertechniek

Inhoud

4/6.1 Microcomputer MPS 65

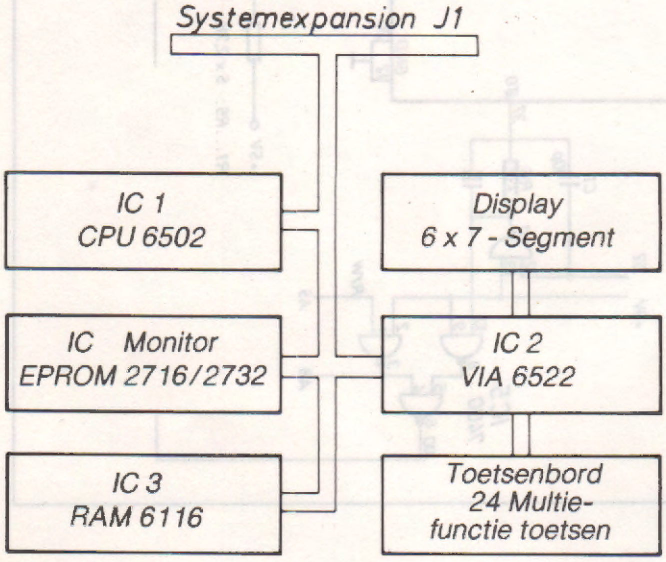
Retroforum

Retroforum

4/6.1

Microcomputer MPS 65

De MPS 65 is een microcomputer, die op een enkele Euro-formaatkaart is gebouwd (een zogenaamde single board computer).
Deze microcomputer kan op vele fronten worden ingezet, denk bijvoorbeeld aan het besturen van een modelbaan, maar ook voor het leren programmeren.
De MPS 65 is een relatief goedkope microcomputer, die zich uitstekend leent om een inzicht te krijgen in principe, opbouw en programmering van microcomputers. Het is een zelfstandig werkend systeem, dat op eenvoudige wijze is uit te breiden met andere systeemfuncties, door middel van op de uitbreidingsbus aan te sluiten Euro-kaarten.



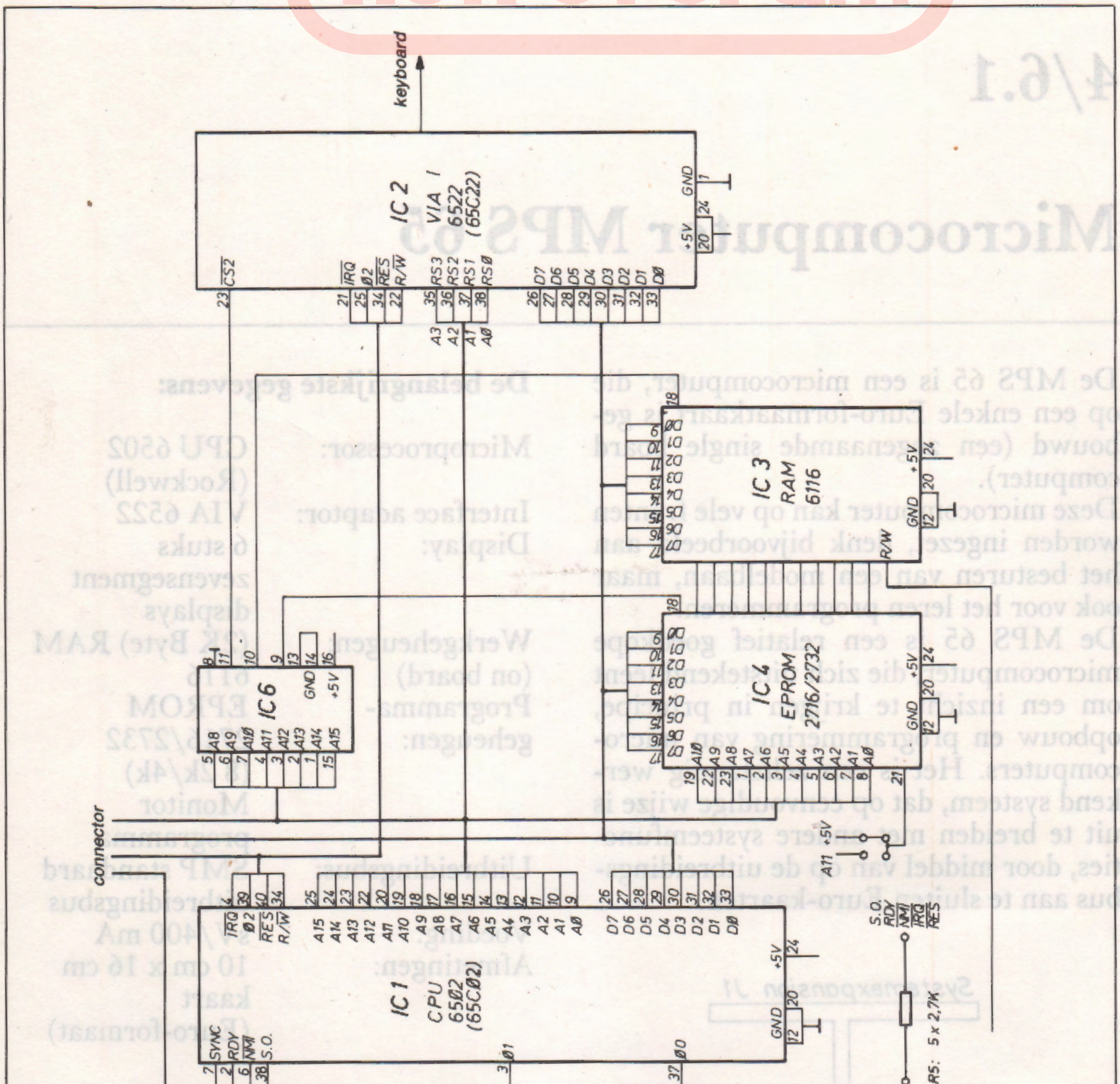
Figuur 4/6.1 -1:
MPS 65 Blokschema

De belangrijkste gegevens:

Microprocessor:	CPU 6502 (Rockwell)
Interface adaptor:	VIA 6522
Display:	6 stuks zevensegment displays
Werkgeheugen: (on board)	(2K Byte) RAM 6116
Programma-geheugen:	EPROM 2716/2732 (8 2k/4k) Monitor programma
Uitbreidingsbus:	SMP standaard uitbreidingsbus
Voeding:	sV/400 mA
Afmetingen:	10 cm x 16 cm kaart (Euro-formaat)

6.1 Microcomputer MPS 65

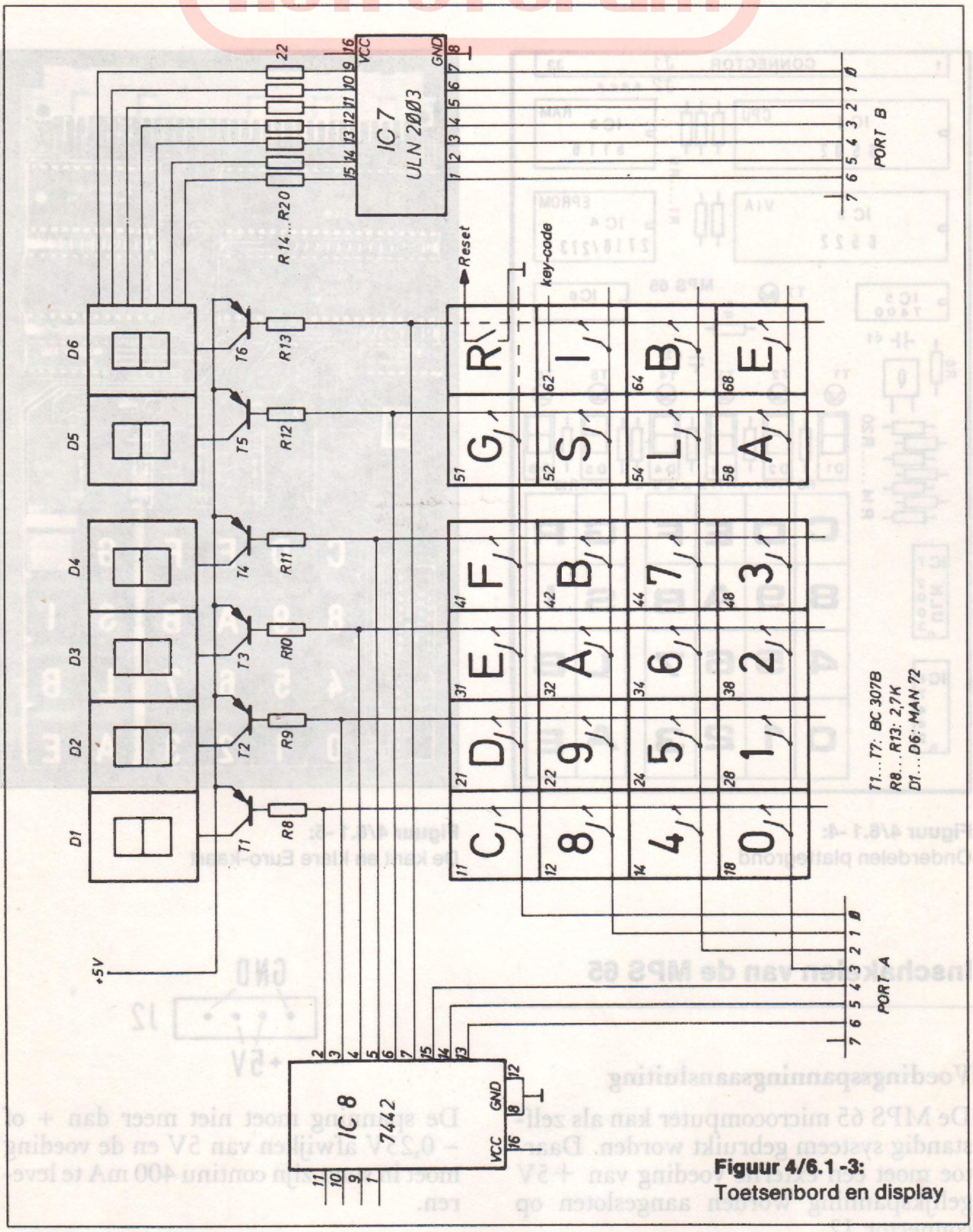
Retroforum



Figuur 4/6.1 -2: Schema

6.1 Microcomputer MPS 65

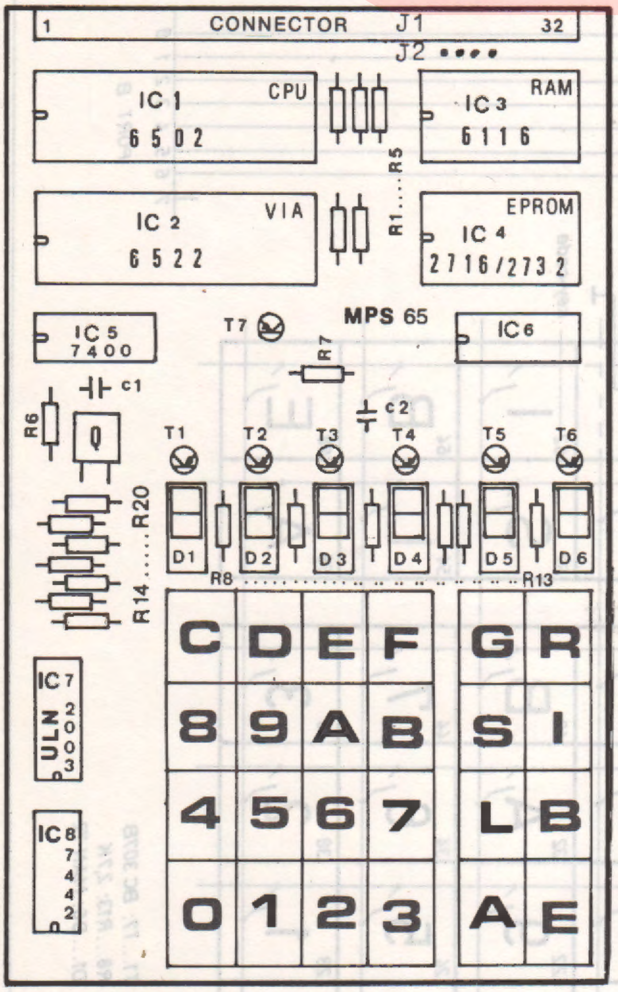
Retroforum



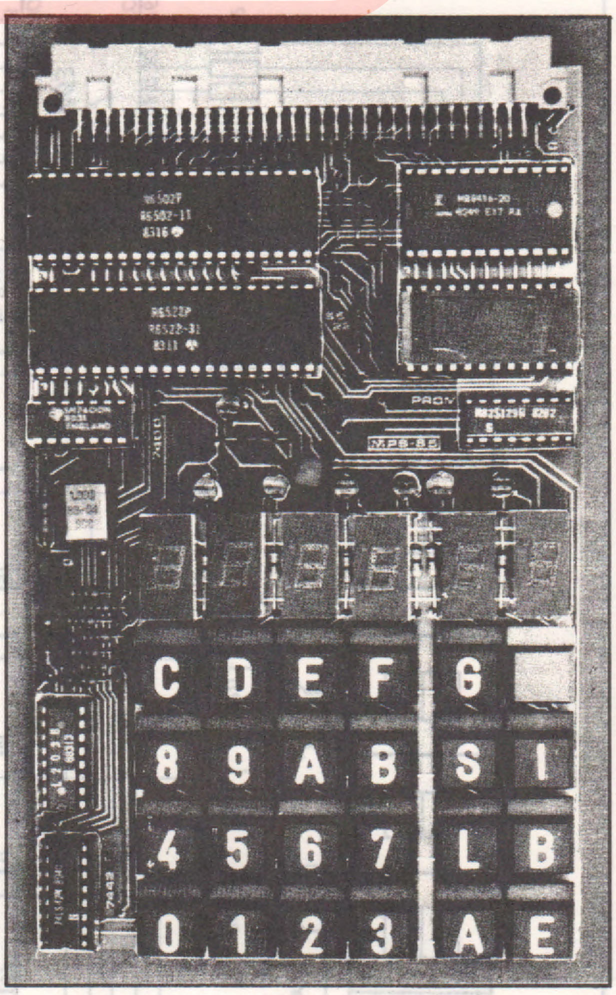
Figuur 4/6.1 -3:
Toetsenbord en display

6.1 Microcomputer MPS 65

Retroforum



Figuur 4/6.1 -4:
Onderdelen plattegrond

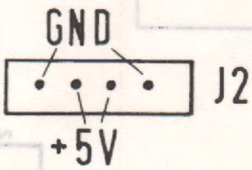


Figuur 4/6.1 -5:
De kant en klare Euro-kaart

Inschakelen van de MPS 65

Voedingsspanningsaansluiting

De MPS 65 microcomputer kan als zelfstandig systeem gebruikt worden. Daartoe moet een externe voeding van +5V gelijkspanning worden aangesloten op connector J2.



De spanning moet niet meer dan + of – 0,25V afwijken van 5V en de voeding moet in staat zijn continu 400 mA te leveren.

6.1 Microcomputer MPS 65

Retroforum

Wat te verwachten valt na het inschakelen!

Zoals van een geciviliseerd apparaat te verwachten is, stelt hij zich voor. Na het inschakelen verschijnt op het display

µPS-65

Na het indrukken van de toets "E" (=enter) verschijnt op het display 0200xx, waarbij xx de inhoud is van de geheugenplaats 0200. Na het inschakelen zal deze inhoud altijd 00 zijn, aangezien het monitorprogramma als eerste actie een geheugenreset uitvoert op het geheugenbereik 0200-07FF.

Opmerking!

1. Alle referenties aan adressen en data in dit hoofdstuk worden in de hexadecimale notatie gedaan, tenzij anders vermeld.
2. Als na inschakelen het display donker blijft, kan door het indrukken van de "R"-toets (Reset) de MPS 65 worden gedwongen opnieuw te starten.

TOETSFUNCTIES (de "keys" van het "keyboard")

Ø-F HEX-toetsen

Hexadecimale toetsen dienen voor de invoer van adressen en data in hexadecimale vorm (hexadecimaal = grondtal 16).
Voor representatie van de hexadecimale eenheden 10–15 wordt gebruik gemaakt van letters, waarbij A = 10, B = 11, C = 12, D = 13, E = 14, F = 15.

Na indrukken van de INDEX-toets krijgen de hexadecimale (kortweg hex) toetsen nog een andere functie. Deze is beschreven onder de beschrijving van de INDEX-toets.

E ENTER-toets

Deze toets heeft verschillende functies, te weten

1. Een stap voorwaarts (single step). Het indrukken heeft tot gevolg, dat het adres met 1 wordt verhoogd (geïncrementeerd). Uiteraard geeft het dataveld dan de inhoud van de nieuwe geheugenplaats weer.
2. Invoer van ingetoetste data in de aangegeven geheugenplaats, waarna de geheugenplaats met 1 wordt verhoogd.
3. Het beëindigen van de monitorprogramma's .
   - Klok op tijd zetten (settime).
   - Tijd aangeven (displaytime).
   - BREAK-functie.

B BACKSTEP-toets

De BACKSTEP-toets heeft de functie, het adres met één te verminderen (decrement), zonder daarbij de in de geheugenplaats aanwezige data te wijzigen.

G GO-toets

Met de GO-toets wordt een gebruikersprogramma gestart, vanaf het op het display aangegeven adres.
Voor het veranderen cq. ingeven van het startadres van een gebruikersprogramma zie A-toets.

6.1 Microcomputer MPS 65

Retroforum

Na het starten van een gebruikersprogramma verschijnt op het display: _ _ _ _ _ _

Voorопgesteld, dat het gebruikersprogramma zelf het display niet gebruikt.

Opmerking:
Het monitorprogramma en de systeemklok werken tijdens de uitvoering van een gebruikersprogramma op interruptbasis. Dat betekent, dat het toetsenbord actief blijft (via de routines KEYIN, KEYIN1) en de displays door het monitorprogramma worden gestuurd (via de adressen ADR16, ADR16+1, DAT8, resp. DISP1 t/m DISP6).

A **ADRES-toets**
Na indrukken van de A-toets worden de vier volgende HEX-toetsen geïnterpreteerd als een nieuw adres. Dit ingetoetste adres en de inhoud hiervan worden op het display zichtbaar, waarna alle andere functies weer beschikbaar zijn.

S **SAVE-toets**
Voor gebruik met uitbreidingskaart(en).

L **LOAD-toets**
Voor gebruik met uitbreidingskaart(en).

I **INDEX-toets**
Door middel van de INDEX-toets krijgt een aantal van de 16 HEX-toetsen een andere functie. Op het display verschijnt:

----In

De speciale functies zijn:

INDEX B	berekenen sprong offset vanaf absolute adressen.
INDEX C	verschuiven van datablok met één plaats om ruimte te maken voor een in te voegen instructie (werkt alleen in geheugenbereik Ø2ØØ-Ø7FF).
INDEX D	display-systeemtijd.
INDEX E	sprong naar, en programmastart vanaf, adres EØØØ.
INDEX F	klok op tijd zetten

BREAK-functie

Een interessante instructie van de 6502 microprocessor is de BREAK-instructie (code ØØ). Zodra de microprocessor een break-instructie detecteert, wordt het lopende programma onderbroken en volgt een sprong (indirect, via adressen FFFE en FFFF) meestal naar een bepaalde routine in het monitorprogramma.
De BREAK-instructie is een software interrupt.
Bij het MPS 65 monitorprogramma, wordt na een BREAK de inhoud van de systeemregisters X, Y en A (Accumulator) weergegeven op het display.

Display: (X-Reg.) (Y-REG.) (ACCU)

De reactie op een BREAK kan echter door de gebruiker worden gewijzigd.

6.1 Microcomputer MPS 65

Retroforum

INDEXFUNCTIES

INDEX F (Systeem)klok op tijd zetten.

Na indrukken van de INDEX-F-toets kan de systeemklok op tijd worden gezet, door met behulp van de toetsen 0-9 het uur en de minuten in te toetsen (in een 24 uurs klok). Na invoer van het vierde digit start de klok automatisch.
Door indrukken van de ENTER-toets verdwijnt het klokdisplay. De klok zelf echter loopt gewoon door. Alleen door de functies RESET, SAVE/LOAD wordt de klok tijdelijk onderbroken, na afloop van deze functies loopt hij echter weer normaal door.
In een gebruikersprogramma kan de tijd worden gebruikt door uitlezing van de adressen ØØA9, ØØAA en ØØAB, waarin respectievelijk de uren, minuten en seconden worden bewaard.

INDEX B Berekenen van de sprongafstand (offset) tussen twee absolute adressen.

Branch-instructies (zoals BEQ, BPL enz.) zijn 2 byte-instructies, waarvan het eerste byte de instructiecode zelf is en het tweede byte de offset tussen het huidige adres en het doeladres.

Bijvoorbeeld:	code	mnemonic
	FØ	BEQ
	Ø9	9 stappen vooruit

Om deze offset op eenvoudige wijze te kunnen berekenen (vooral bij negatieve sprongen) kan de INDEX B-functie gebruikt worden. De instructie zelf bijv. BEQ = FØ wordt normaal ingetoetst en met ENTER ingevoerd.
Op de plaats waar de offset moet worden ingevoerd, wordt dan INDEX B gedrukt, waarna het absolute doeladres (in vier digits) wordt ingetoetst. Automatisch wordt dan de offset berekend, die nu met ENTER wordt ingevoerd.

Voorbeeld: Ø22Ø FØ ØE BEQ Ø23Ø
.
.
Ø23Ø Hier wordt in plaats van OE ingevoerd:
INDEX/B/Ø23Ø/ENTER

INDEX C Data-blok verschuiven in geheugenbereik Ø2ØØ-Ø7FF.
Moet in een programma een instructie (of deel daarvan) worden ingevoegd, dan kan met behulp van INDEX C het gehele programma vanaf de op het display aangegeven geheugenplaats één plaats

Retroforum

worden opgeschoven. De open plaats die ontstaat wordt automatisch met een NOP = EA instructie gevuld, die echter daarna overgeschreven kan worden.
NB: Eventueel in het programma voorkomende absolute adresreferenties, die door de verschuiving niet meer kloppen worden *niet* automatisch aangepast.

INDEX D

Zet de lopende systeemtijd op het display, deze functie kan met de ENTER-toets worden beëindigd.

INDEX E

Het systeem voert een sprong uit naar adres EØØØ.
Een op adres E005 beginnend programma wordt door het systeem na initialisering gestart, als op de geheugenplaatsen EØØ3 en EØØ4 de codes Ø1 resp. FF staan.

Dus:	EØØ3	Ø1	
	EØØ4	FF	
	EØØ5		Startadres

De geheugenplaatsen EØØØ tot EØØ2 worden vrijgehouden, om daarvandaan met een JMP-instructie naar het begin van het programma te kunnen springen.
Toepassing: INDEX E.

FOUTMELDINGEN

Het MPS 65 systeem kan verschillende foutmeldingen geven, die met een nummer worden aangeduid.

Deze nummers hebben de volgende betekenis:

ERROR 1 Foutieve adresinvoer (beginadres > eindadres)
ERROR 2 Relatieve sprong te ver terug. Doeladres ligt meer dan 128 plaatsen terug.
ERROR 3 Relatieve sprong te ver naar voren. Doeladres ligt meer dan 127 plaatsen verder.
ERROR 4 Foutieve branch-instructie.
ERROR 6 Fout bij de afhandeling van interruptvectoren.

Na het indrukken van de ENTER-toets is een fout "gereset" en kan de correcte data ingetoetst worden.

6.1 Microcomputer MPS 65

Retroforum

Tabel 1: Monitorroutines van de MPS 65

Naam	Adres	Register	Beschrijving
KEYIN	F800	A	Uitlezen van toetsenbord met tijdsvertraging, de toetscode staat in de accu (contactdenden onderdrukt!)
KEYIN1	F803	A	Uitlezen van toetsenbord, zonder tijdsvertraging, de toetscode staat in de accu.
DISPEX	F806		Inschakelen van externe displaymode (voor besturing van de individuele elementen van de displays.
DISPSY	F809		Inschakelen van system displaymode (voor hexadecimale representatie van de "cijfers" 0-F.
PHAXY	F80C		Bewaren van de inhoud van Accu, X- en Y-register op de stack.
PLAXY	F80F	A,X,Y	Herstellen van de waarden van Accu, X- en Y-register met op de stack bewaarde (oude) inhouden.
ASK2	F812	A,X	Inlezen van een twee digits groot hexadecimaal getal. Het getal staat in adres 00A0.
ASK4	F815	A,X	Idem ASK2, maar voor 4 digits. Het getal staat in 00A0 en 00A1.
MON65	F818		Startadres van het MONITOR-programma van de MPS65, zonder initialisatie.
MON 65E	F81B		Startadres van het MONITOR-programma, waar naar toe wordt gesprongen na indrukken van de ENTER-toets.
T01S	F81E		Vertragingsroutine. Duur van de tijdsvertraging is 0,1 x de waarde in de accu in sec.
INC16X	F821		Een zestienbits getal wordt geïncrementeerd (met 1 verhoogd).
DEC16X	F824		Een zestienbits getal wordt gedecrementeerd (met 1 verlaagd).
ERR	F827	A,X,Y	Displaymelding ERROR (A). (A) is de waarde van de ACCU van 0-F.
STRICH	F82D		Display: ------
SUCHEX	F830	A	Uitlezing van het toetsenbord, met tijdsvertraging (de gedrukte toets staat in de Accu – 0-F).

6.1 Microcomputer MPS 65

Retroforum

Systeemroutines (beschrijving)

Een aantal van de monitorroutines kunnen door de gebruiker als subroutine worden aangeroepen in zijn programma's.

Veel gebruikte systeemfuncties kunnen door gebruikmaking van monitorroutines eenvoudig worden geïmplementeerd, waarbij men dan de zekerheid heeft dat daarin geen fouten zitten, terwijl deze routines in ieder geval met de systeemtiming rekening houden.

Voor de meeste routines fungeert de accumulator als parameteroverdrachtsregister (bijv. code van de ingedrukte toets bij routine KEYIN). X en Y register blijven, voorzover zij niet bij de parameteroverdracht zijn betrokken ongewijzigd (met uitzondering van ERR-F827).

Het monitorprogramma staat in PROM (programmable read only memory = niet vluchtig geheugen) vanaf adres F8ØØ. De routines worden door middel van een JSR-instructie aangeroepen.

De subroutineaanroepen worden via een sprongtabel (vanaf F8ØØ) naar het juiste startadres van de subroutines gedirigeerd (een soort indirecte subroutine aanroep).

Deze procedure heeft enkele voordelen, we noemen:

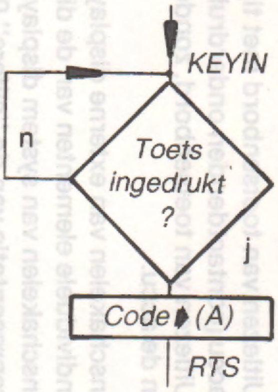
- de aanroepadressen liggen achter elkaar en zijn gemakkelijk te onthouden.
- de aanroepadressen veranderen niet bij aanpassing of verandering van het monitorprogramma.

Daardoor behoeven gebruikersprogramma's niet te worden aangepast bij wijziging van het monitorprogramma.

= = KEYIN = =

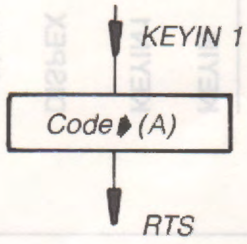
De aanroep van de routine KEYIN heeft tot gevolg, dat het systeem wacht tot er een toets ingedrukt wordt, dan wordt de code van deze toets in de accu gezet, waarna de terugkeer naar het hoofdprogramma volgt.

Verandering in register(s): geen



= = KEYIN 1 = =

De werking van deze subroutine is nagenoeg gelijk aan die van de vorige, zij het dan dat er niet wordt gewacht tot een toets ingedrukt is. (Voor toetscode zie tabel 2).

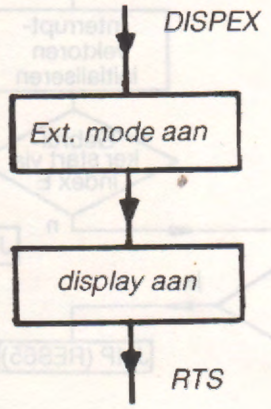


Retroforum

= = DISPEX = =

Routine voor het omschakelen van de displaymode in externe aansturing. Hierbij staan de codes (het bit-patroon) voor elk van de 6 displays in geheugenlocaties DISP 1 . . . DISP6). (Voor bitpatronen en juiste adressen zie figuur 8)
Bij het starten van een programma door middel van de "GO"-toets wordt deze mode automatisch ingeschakeld.
DISPEX is dus alleen nodig als externe displaybesturing nodig is, nadat het gebruikersprogramma zelf via DISPSY de mode veranderd had.
DISPEX schakelt bovendien de displays weer aan, voor het geval zij uit waren.

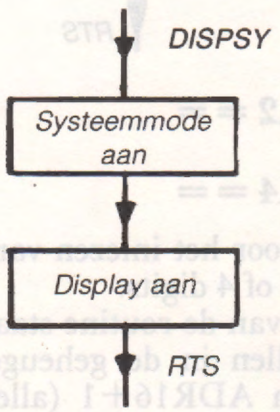
Verandering in register(s): geen



= = DISPSY = =

Routine voor het omschakelen van de externe displaymode in systeembesturing.
Bij systeembesturing wordt de in de adressen ADR16, ADR16+1 en DAT8 staande data als hexadecimale waarde van respectievelijk telkens linker- en rechter-nibble (nibble = $\frac{1}{2}$ byte) dus 6 in totaal) in de displays 1 . . . 6 weergegeven. De subroutine verzorgt dus zelf de aansturing van de benodigde bits. DISPSY schakelt bovendien de displays aan indien zij uitgeschakeld waren.

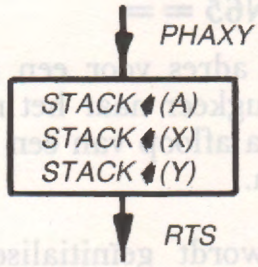
Verandering in register(s): geen



= = PHAXY = =

Deze routine bewaart de inhouden van accumulator, het X-register en het Y-register op de stack.

Verandering in register(s): geen



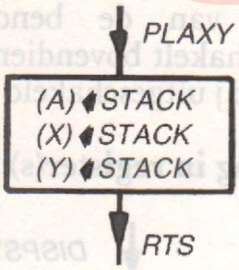
= = PLAXY = =

Met deze routine kunnen de met PHAXY op de stack bewaarde registerinhouden weer in de betreffende registers worden teruggelezen.

Verandering in register(s): Accu, X en Y.

6.1 Microcomputer MPS 65

Retroforum



== ASK2 ==

== ASK4 ==

Routines voor het inlezen van een getal van resp. 2 of 4 digits.
Na afloop van de routine staan de ingelezen getallen in de geheugenplaatsen ADR16 en ADR16+1 (alleen gevuld door ASK4).

Verandering in register(s): A, X

N.B. De inhoud van geheugenplaatsen ADR16 en ADR16+1 worden door het systeem overschreven na een RESET, of na een JMP MON65 instructie.

== MON65 ==

Dit is een adres voor een geprogrammeerde terugkeer naar het monitorprogramma, na afloop van een gebruikersprogramma.

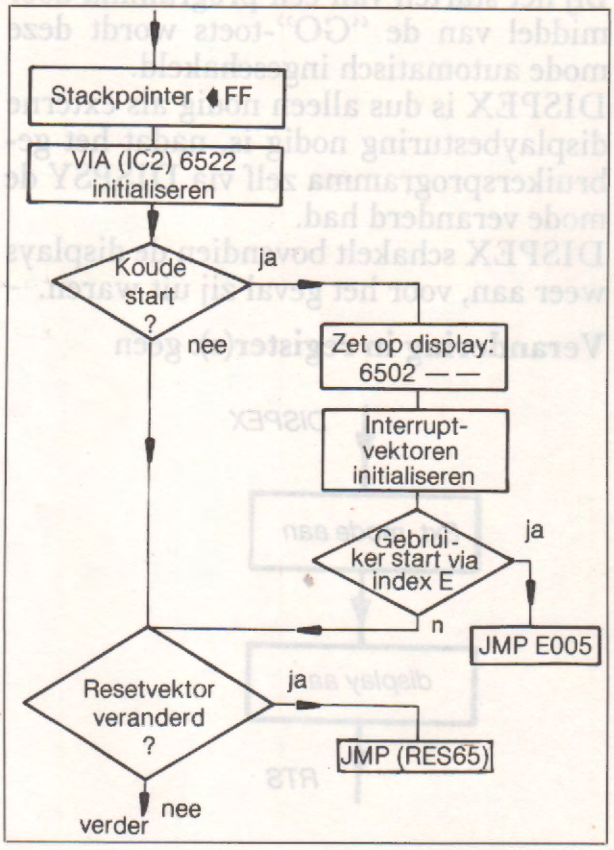
De VIA wordt geïnitialiseerd en de stackpointer wordt gereset (op FF gezet).

Toepassing:
Beëindiging van een gebruikersprogramma door gebruik te maken van JMP MON65.

Gebruikersprogramma-instructies:

:

4C 18 F8 JMP MON65



Figuur 6:
Initialisering

6.1 Microcomputer MPS 65

Retroforum

= = MON 65E = =

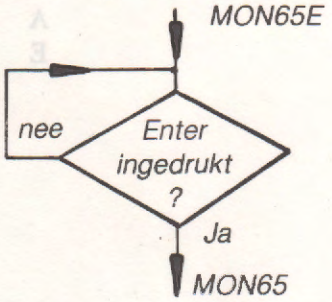
Beëindigen van een gebruikersprogramma als boven, echter de laatste door het gebruikersprogramma op de displays gezette informatie mag niet worden overschreden.
Dit kan worden bereikt, door in plaats van een JMP MON65 een JMP MON65E uit te voeren als laatste instructie van het gebruikersprogramma.

De monitor wacht nu op het indrukken van de ENTER-toets, alvorens de re-initialisatie uit te voeren en de displays weer te gebruiken voor adres en data-display van de monitor zelf.

Gebruikersprogramma-instructies:

.
.

4C 1B F8 JMP MON65E



= = TØ1S = =

Dit is een delay(tijdvertragings)-routine.
Aanroep van deze routine heeft tot gevolg, dat de terugkeer naar het hoofdprogramma (0,1 maal de inhoud van de accu) seconden duurt. Dat wil zeggen dat er tijdvertragingen van 0,1 tot 25,5 seconden (accu-inhoud hex Ø1 tot FF) mogelijk zijn.
Bijv.: Als bij aanroep van de routine de accu hex Ø9 bevat, zal de terugkeer naar het hoofdprogramma (en dus de tijdvertraging) 0,9 seconden duren.
Dit voorbeeld kan als volgt worden geprogrammeerd:
instructies:

A9 Ø9	LFA	#$Ø9;	laad de accu met hex Ø9
2Ø 1E F8	JSR	TØ1S:	roep vertragingsroutine aan
			instructies

= = INC16X = =

Deze routine incrementeert (= verhoogd met 1) een zestien bits getal. Het getal moet op pagina 0 staan en het adres in deze pagina moet in het X-register staan.

Voorbeeld:

Op adressen ØØ35 en ØØ36 staat het zestien bits getal 34FF, dat geïncrementeerd moet worden.

adres	data
ØØ35	FF
ØØ36	34

Na uitvoering van

formaat:

A2 35	LDX	#$35;	laad X-register met hex 35
20 21F8	JSR	INC 16X;	incrementeer 16 bit data

is de data: ØØ35 ØØ
ØØ36 35

Zoals bij de 6502-processor gebruikelijk staat het minst significante byte (het rechter byte) voorop, gevolgd door het meest significante byte (het linker byte).

Verandering in register(s): geen

Retroforum

= = DEC16X = =

Het was te verwachten, dat als er een incrementeer-routine is, dat er ook een decrementeer-routine bestaat. Decrementeren = met 1 verlagen. Deze routine werkt overeenkomstig INC16X.

Verandering in register(s): geen

= = ERR = =

Routine voor foutmelding met foutcode (bijv. ERROR 3).

De foutcode moet voor de aanroep van de subroutine in de accu worden gezet. Er zijn zestien foutcodes mogelijk: ØØ tot 0F.

Voorbeeld om op display ERROR 3 te zetten.

```
A9  Ø3      LDA   #$Ø3
2Ø  27 F8   TSR   ERR
```

N.B.: De subroutine ERR schakelt de displaymode op DISPSY.

Verandering in register(s): A, X

Code	hex toets	Code	funktietoets
18	0	51	G
28	1	52	S
38	2	62	I
48	3	54	L
14	4	64	B
24	5	58	A
34	6	68	E
44	7		
12	8		
22	9		
32	A		
42	B		
11	C		
21	D		
31	E		
41	F		

Tabel 4/6.1 -2:
De toetsencodes van de MPS 65

Adressen in pagina 0, die door de monitor worden gebruikt zijn te vinden in tabel 3.

6.1 Microcomputer MPS 65

Retroforum

ØØAØ	= ADR16		
ØØA1	= ADR16+1		
ØØA2	= DAT8		
ØØA3	= DISP1		
ØØA4	= DISP2		
ØØA5	= DISP3		
ØØA6	= DISP4		
ØØA7	= DISP5		
ØØA8	= DISP6		
ØØA9	= STD		
ØØAA	= MIN		
ØØAB	= SEK		
ØØAD	= GA		
ØØAE	= GX		
ØØAF	= GY		
ØØCØ	= RESF		
ØØC1	= RES65	(L)	
ØØC2	= RES65	(H)	

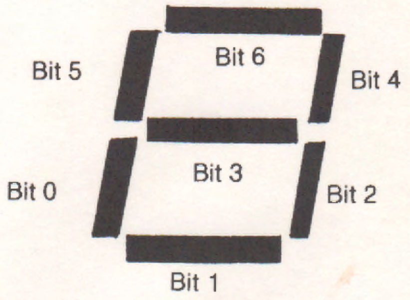
ØØC3	= IRQF	
ØØC4	= IRQ65	(L)
ØØC5	= IRQ65	(H)
ØØC6	= NMIF	
ØØC7	= NMI65	(L)
ØØC8	= NMI65	(H)
ØØC9	= IRQMF	
ØØCA	= IRQM	(L)
ØØCB	= IRQM	(H)
ØØCC	= BRKUF	
ØØCD	= BRKU	(L)
ØØCE	= BRKU	(H)

Tabel 4/6.1 -3
De zero-page adressen.

Aansluitingen van de zevensegment displays:

MODE						
DISPSY	DISP 1	DISP 2	DISP 3	DISP 4	DISP 5	DISP 6
DISPEX	ADR 16 + 1		ADR 16		DAT 8	

Figuur 4/6.1 -7
De monitoradressen van het display.



Figuur 4/6.1 -8
Displayaansturing.

Retroforum

Voorbeeld om ABCDEF op het display te zetten:

	bits	binair	hex.
A	0,2,3,4,5,6	0111 1101	7D
B	0,1,2,3,4,5,6	0111 1111	7F
C	0,1,5,6	0110 0011	63
D	0,1,2,4,5,6	0111 0111	77
E	0,1,3,5,6	0110 1011	6B
F	0,3,5,6	0110 1001	69

Door de gevonden hexadecimale getallen in de geheugenplaatsen DISP1 ... DISP6 te zetten en dan de routine DISPEX aan te roepen, zullen de letters A . . . F op de zes displays verschijnen.

Interrupt afhandeling!

De afhandeling van optredende interrupts door de MPS 65.

Het systeem kent de volgende interrupts:

vectoradres	vector
RES65	reset vector
IRQ65	IRQ (hardware; maskable) vector
NMI65	NMI (hardware; non-maskable) vector
BRKU	breakvector (software interrupt) voor gebruiker (met monitorprogramma)
IRQM	IRQ-vector voor gebruiker (met monitorprogramma); wordt automatisch door de geprogrammeerde timer geactiveerd (elke 2,5 ms)

Bij gebruik van de vectoren door de gebruikers, moeten de volgende regels strikt in acht worden genomen:

1. Eerst het adres van het gebruikers-interruptprogramma in de vectorgeheugenplaatsen zetten.
2. De bij deze vector behorende vlag op nul zetten (d.w.z. de geheugenplaats die als vlag wordt gebruikt met de waarde 00 opvullen). Het vlagadres van elke vector is de geheugenplaats voor de betreffende vector.

6.1 Microcomputer MPS 65

Retroforum

De verschillende vectorafhandelingen **moeten** het volgende formaat hebben:

RES65
De vector wijst naar het aanvangsadres van het programma, dat gestart moet worden na indrukken van de RESET-toets.

```
IRQ65   .
        .
        (Programma)
        .
        JSR PLAXY
        RTI
```

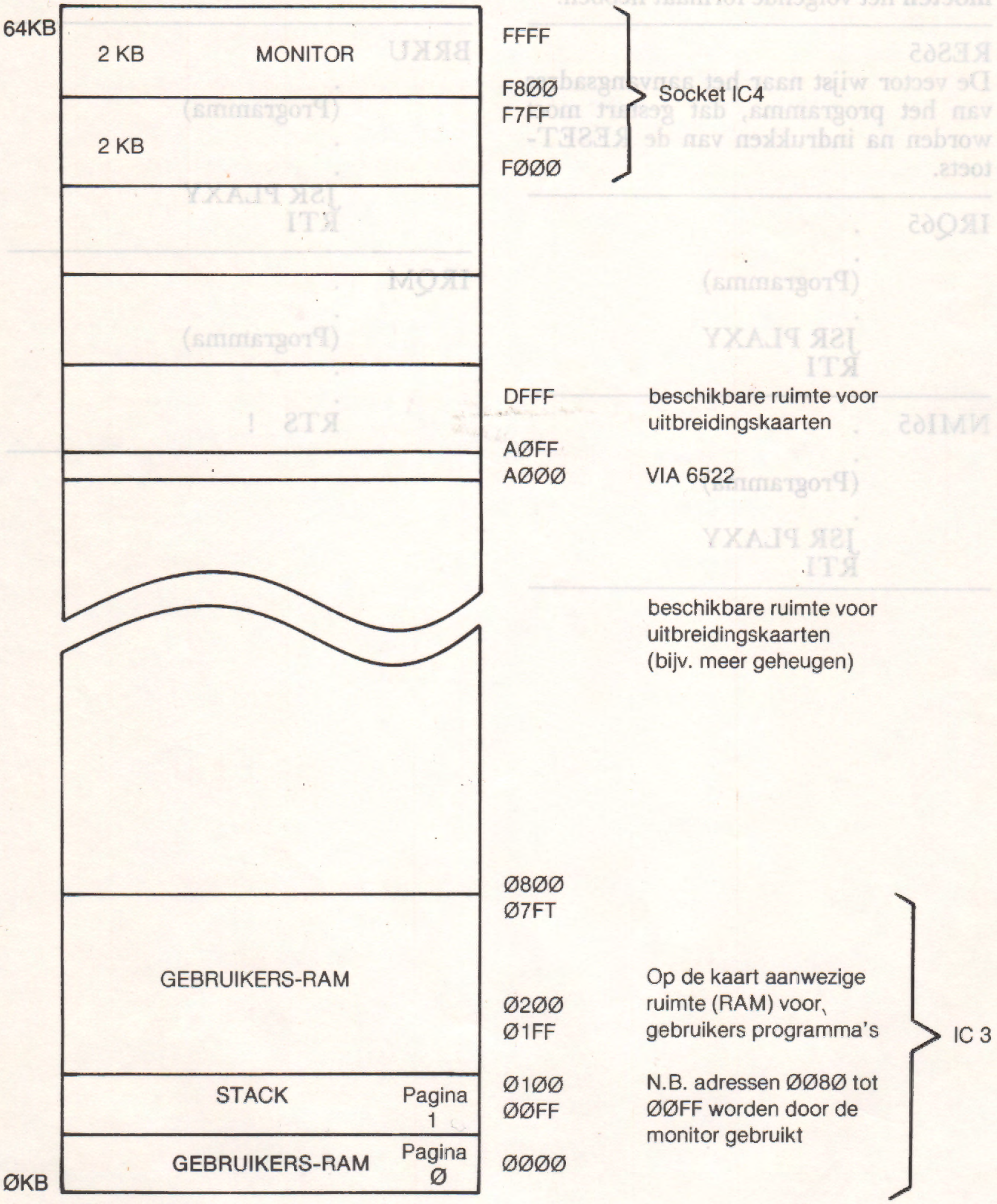
```
NMI65   .
        .
        (Programma)
        .
        JSR PLAXY
        RTI
```

```
BRKU    .
        .
        (Programma)
        .
        .
        JSR PLAXY
        RTI
```

```
IRQM    .
        .
        (Programma)
        .
        .
        RTS   !
```

Retroforum

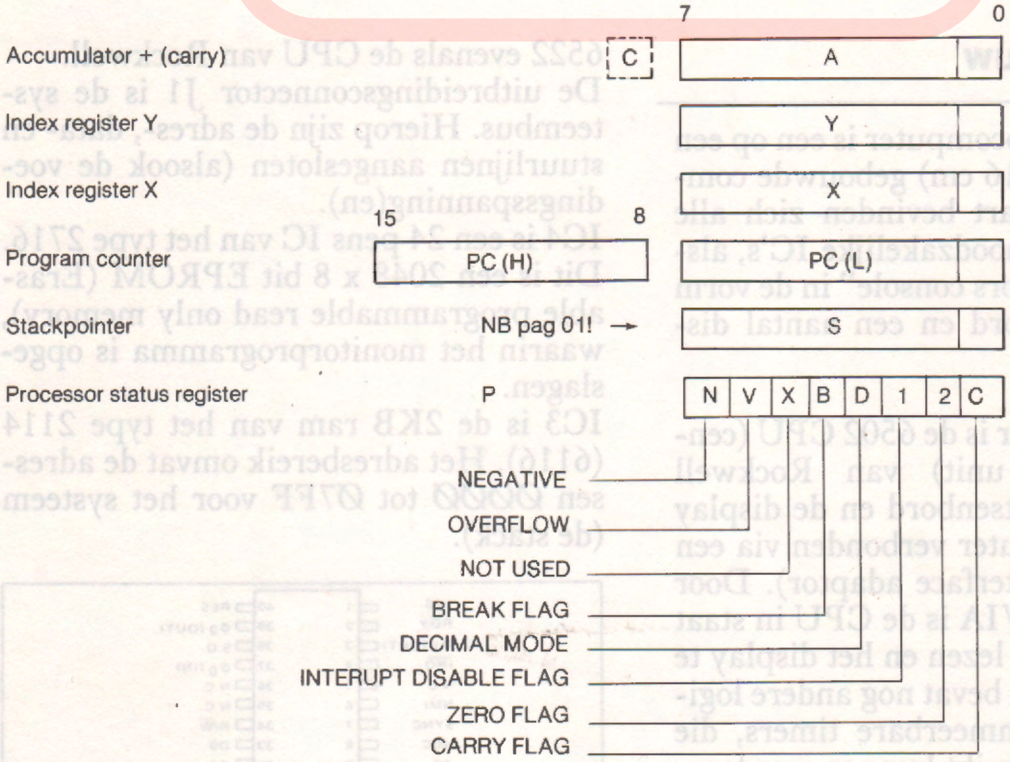
MPS 65 Geheugenindeling



Figuur 4/6.1 -9:
MPS 65 Geheugenindeling.

6.1 Microcomputer MPS 65

Retroforum



Figuur 4/6.1 -10:
De 6502 registers.

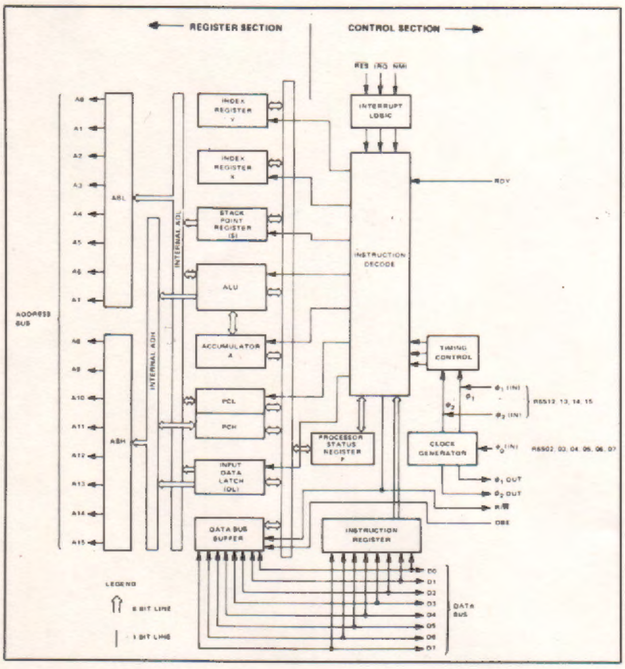
6.1 Microcomputer MPS 65

Retroforum

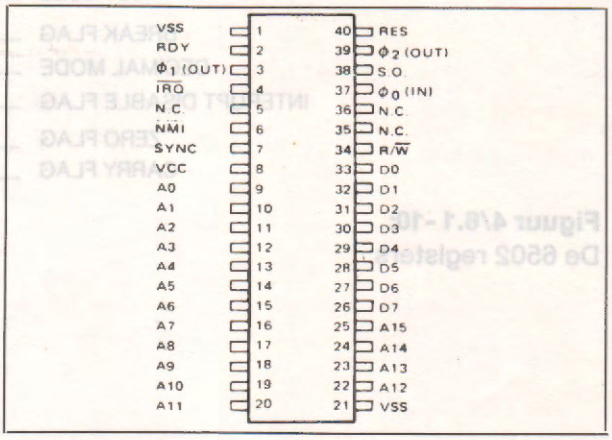
Systeemopbouw

De MPS 65 microcomputer is een op een Eurokaart (10 x 16 cm) gebouwde computer. Op de kaart bevinden zich alle voor de werking noodzakelijke IC's, alsmede een "operators console" in de vorm van een toetsenbord en een aantal displays.

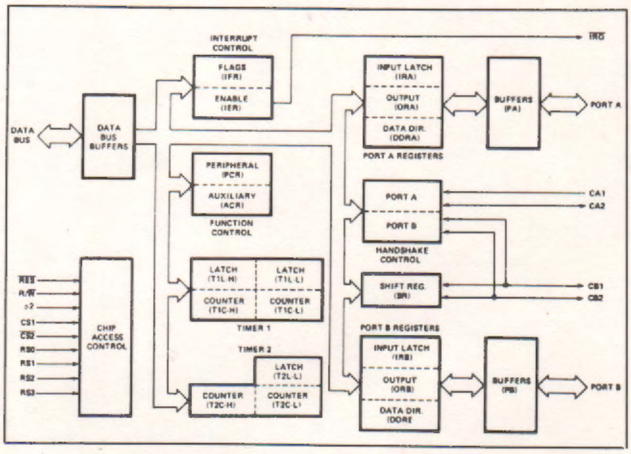
Als microprocessor is de 6502 CPU (central processing unit) van Rockwell gebruikt. Het toetsenbord en de display zijn met de computer verbonden via een VIA (versatile interface adaptor). Door middel van deze VIA is de CPU in staat het toetsenbord te lezen en het display te besturen. De VIA bevat nog andere logica, zoals programmeerbare timers, die door de CPU gebruikt kunnen worden.
De VIA is "familie" van de CPU. De op de MPS 65 gebruikte VIA is het type 6522 evenals de CPU van Rockwell.
De uitbreidingsconnector J1 is de systeembus. Hierop zijn de adres-, data- en stuurlijnen aangesloten (alsook de voedingsspanning(en).
IC4 is een 24 pens IC van het type 2716. Dit is een 2048 x 8 bit EPROM (Erasable programmable read only memory), waarin het monitorprogramma is opgeslagen.
IC3 is de 2KB ram van het type 2114 (6116). Het adresbereik omvat de adressen ØØØØ tot Ø7FF voor het systeem (de stack).



Figuur 4/6.1 -11:
Interne opbouw van de 6502 CPU.



Figuur 4/6.1 -12:
De aansluitingen van de 6502.



Figuur 4/6.1 -13:
Blokschema van de R6522 VIA.

6.1 Microcomputer MPS 65

Retroforum

De belangrijkste technische gegevens van de VIA zijn:

1. Twee 8 bits-bidirectionele ingangs/uitgangspoorten.
2. Twee 16 bits programmeerbare timers/counters.
3. Seriële ingangspoort.
4. Enkelvoudige voeding (+ 5V).
5. TTL-compatibel.
6. CMOS-compatibele controlelijnen.
7. 1 en 2 MHz klokfrequentie mogelijk.

6502 Adresseermogelijkheden

(de vetgedrukte data staat na uitvoering van de instructie in de accu)

IMMEDIATE (IMM)

LDA #$AA	0200	A9	Instructie
	0201	**AA**	data

ABSOLUTE (ABS)

LDA $0312	0200	AD	Instructie
	0201	12	adresbyte (laag)
	0202	03	adresbyte (hoog)
	0312	**AA**	data

ZERO PAGE (Z)

LDA (Z)-$BA	00BA	**AA**	data
	0200	A5	instructie
	0201	BA	adresbyte (laag!)

ABSOLUTE INDEXED (X) (ABS,X)

LDA, X	0200	BD	instructie
	0201	01	adresbyte (laag)
(stel X=5)	0202	03	adresbyte (hoog)
	0301		doeladres
		+X	offset
	0306	**AA**	data

ABSOLUTE INDEXED (Y) (ABS, Y)
als boven met Y-register in plaats van X-register

ZERO PAGE INDEXED (X) (Z, X)

0020

stel X=5

6.1 Microcomputer MPS 65

Retroforum

		+ X	offset
	0025	**AA**	data
LDA(Z), X	0200		instructie
	0201	20	adresbyte (laag!)

ZERO PAGE INDEXED (Y) (Z, Y)
als boven met Y-register in plaats van X-register

INDEXED, X INDIRECT (IND, X)

	0020		
		+ X	offset
stel X=5	0025	12	adresbyte (laag)
	0026	03	adresbyte (hoog)
LDA ($20, X)	0200	A1	instructie
	0201	20	adresbyte (laag!)
	0312	**AA**	data

INDIRECT Y-INDEXED ((IND), Y)

stel Y=5	0020	12	adresbyte (laag)
	0021	03	adresbyte (hoog)
LDA ($20), Y	0200	B1	instructie
	0201	20	adresbyte (laag)
	0312	--	
		+ Y	offset
	0317	**AA**	data

INDIRECT JUMP

JMP ($0212)	0200	6C	instructie
	0201	12	adresbyte (laag)
	0202	02	adresbyte (hoog)
	0212	02	adresbyte (laag)
	0213	03	adresbyte (hoog)
	0302		Na de uitvoering van de instructie gaat het programma vanaf geheugenplaats 0302 verder.

6.1 Microcomputer MPS 65

Retroforum

Memory referenced instructies

	IMM	ABS	Z	ABS, X	ABS, Y	Z, X	Z, Y	(IND,) X	(IND), Y	IND
ADC	69	6D	65	7D	79	75		61	71	
AND	29	2D	25	3D	39	35		21	31	
ASL		ØE	Ø6	1E		16				
BIT		2C	24							
CMP	C9	CD	C5	DD	D9	D5		C1	D1	
CPX	EØ	EC	E4							
CPY	CØ	CC	C4							
DEC		CE	C6	DE		D6				
EOR	49	4D	45	5D	59	55		41	51	
INC		EE	E6	FE		F6				
JMP		4C								6C
JSR		20								
LDA	A9	AD	A5	BD	B9	B5		A1	B1	
LDX	A2	AE	A6		BE		B6			
LDY	AØ	AC	A4	BC		B4				
LSR		4E	46	5E		56				
ORA	Ø9	ØD	Ø5	1D	19	15		Ø1	11	
ROL		2E	26	3E		36				
ROR		6E	66	7E		76				
SBC	E9	ED	E5	FD	F9	F5		E1	F1	
STA		8D	85	9D	99	95		81	91	
STX		8E	86				96			
STY		8C	84			94				

Implied instructies

BRK	ØØ	PLP	28
CLC	18	RTI	4Ø
CLD	D8	RTS	6Ø
CLI	58	SEC	38
CLV	B8	SED	F8
DEX	CA	SEI	78
DEY	88	TAX	AA
INX	E8	TAY	A8
INY	C8	TSX	BA
NOP	EA	TXA	8A
PHA	48	TXS	9A
PHP	Ø8	TYA	98
PLA	68		

Instr.	code	vlag status
BCC	9Ø	C = Ø
BCS	BØ	C = 1
BEQ	FØ	Z = 1
BMI	3Ø	N = 1
BNE	DØ	Z = Ø
BPL	1Ø	N = Ø
BVC	5Ø	V = Ø
BVS	7Ø	V = 1

(Zie processor status register voor vlaggen)

Tabel 4/6.1 -4: 6502 Instructieset.

Retroforum

Programmavoorbeelden

Opdracht 1:

Op displays 1 en 5 moet het teken "L" worden weergegeven.

"L" is geen hexadecimaal teken, daarom kan dit alleen worden bereikt, door via externe aansturing de betreffende displaysegmenten te activeren. Dit gebeurt in de mode DISPEX, die normaal door het monitorprogramma wordt ingeschakeld wanneer naar een gebruikersprogramma wordt gesprongen.

Voor het teken "L" moeten de segmenten 0, 1 en 5 oplichten (zie fig. 8), hetgeen overeenkomt met hexadecimaal 23.

Om het monitorprogramma een "L" op display 1 en 5 te laten zetten, moet de code 23 op adressen DISP1 en DISP5 (ØØA3 en ØØA7) worden gezet.

Met een sprong naar MON65E wordt het gebruikersprogramma beëindigd.

Het programma:		adres:	hex-code:
LDA	#$23	0200	A9
		0201	23
STA	DISP1	0202	85
		0203	A3
STA	DISP5	0204	85
		0205	A7
JMP	MON65E	0206	4C
		0207	1B
		0208	F8

Voer het programma in (de hex-code) door elke hex-code in te voeren, gevolgd door de ENTER-toets.
Daarna met behulp van de ADRES-toets het startadres (0200) invoeren.

Het programma wordt gestart met de GO-toets.

Het display moet nu het volgende plaatje te zien geven:

L--- L-

Met behulp van de ENTER-toets kan men nu weer terugkeren naar de monitor.

Opdracht 2:

Op de displays moeten willekeurige segmenten aangestuurd kunnen worden. De codes daarvan staan in een tabel, TAB, die begint op geheugenplaats ØØØØ.

(De BPL sprongafstand wordt met behulp van de INDEX B-toets bepaald!)

Het programma:		adres:	hex-code:
	LDX #$05	0200	A2
		0201	05
HERH:	LDA TAB, X	0202	B5
		0203	00
	STA DISP1, X	0204	95
		0205	A3
	DEX	0206	CA
	BPL HERH	0207	10
		0208	F9 (VIA INDEX B)
	JMP MON65E	020A	4C
		020B	1B
		020C	F8

De zes tekencodes, die weergegeven moeten worden, worden in adressen ØØØØ-ØØØ5 gezet.

Bijvoorbeeld:

HALLO

6.1 Microcomputer MPS 65

Retroforum

```
TAB:  0000   3D ▶   00A3   DISP1
      0001   7D ▶   00A4   DISP2
      0002   23 ▶   00A5   DISP3
      0003   25 ▶   00A6   DISP4
      0004   77 ▶   00A7   DISP5
      0005   08 ▶   00A8   DISP6
```

Start het programma als bij opdracht 1.

Opdracht 3:

Toon een ingetoetste (hexadecimale) code op displays 5 en 6.

Aangezien de start van een gebruikersprogramma vanuit de monitor altijd het inschakelen van de DISPEX-mode tot gevolg heeft, is de eerste taak die het gebruikersprogramma moet uitvoeren, het terugzetten in de hexadecimale displaymode (DISPSY).

Het toetsenbord wordt met de monitorsubroutine KEYIN gelezen, de waarde van de accu (de toetswaarde) wordt dan in DATS gezet, waarna het monitorprogramma de code displayed.

```
Het programma:          adres:  hex-code:

      JSR  DISPSY       0200    20
                        0201    09
                        0202    F8
HERH: JSR  KEYIN        0203    20
                        0204    00
                        0205    F8
      STA  DATS         0206    85
                        0207    A2
      JMP  HERH         0208    4C
                        0209    03
                        020A    02
```

Start het programma als bij opdracht 1. Dit programma zal telkens na indrukken van een toets de toetswaarde in displays 5 en 6 tonen, behalve van de RESET-toets (zie tabel 2). De RESET-toets is dan ook de enige methode om het programma te onderbreken en terug te komen naar de monitor.

Opdracht 4:

Verhoog de waarde op display 1 en 2 voortdurend met 1 (incrementeren)

Er wordt bij dit programma gebruik gemaakt van de tijdvertragingsroutine, om de diverse waarden op de display ook zichtbaar te maken.

```
Het programma:          adres:  hex-code

      JSR  DISPSY       0200    20
                        0201    09
                        0202    F8
HERH: LDA  #$03         0203    A9
                        0204    03 ; 3x
      JSR  TØ1S         0205    20 ; 0,1 sec.
                        0206    1E
                        0207    F8
      INC  A1           0208    E6 ; Increment
                        0209    A1 ; DATS
      JMP  HERH         020A    4C
                        020B    03
                        020C    02
```

Opdracht 5:

Start een programma op een bepaalde tevoren ingegeven tijd.

Een singleboard computer als de MPS 65 kan eenvoudig worden uitgebreid met elektronica om er een schakelklok van te maken.
Bij deze opdracht wordt aangetoond hoe het programma, met gebruikmaking van de interne, onder de monitor lopende klok een ander in het geheugen aanwezig programma kan starten.

6.1 Microcomputer MPS 65

Retroforum

Er wordt vanuitgegaan, dat de schakeltijd in de geheugenplaatsen ØØ, Ø1 en Ø2 van pagina nul staan. Van de lopende klok staat de tijd in de geheugenplaatsen STD, MIN, SEC op de adressen A9, AA en AB van pagina nul.

Het programma:	adres:	hex-code:
LDX #$02	0200	A2
	0201	02
LDA 00, X	0202	B5 ; haal schakeltijd op
	0203	00
CMP STD, X	0204	D5 ; vergelijk met kloktijd
	0205	A9
BNE $200	0206	D0 ; opnieuw indien niet
	0207	F8 ; gelijk
DEX	0208	CA ; nu min en uren
BPL $202	0209	10
	020A	F7
	020B	; hier begint het te starten programma.

Het programma dat in het geheugen staat vanaf adres 020B zal worden uitgevoerd, zodra de lopende klok in adressen A9 t/m AB gelijk is aan de schakeltijd in de adressen ØØ t/m Ø1. Met behulp van dit programma kunnen dan de gewenste functies worden uitgevoerd.

Het starten van programma's met gedefinieerde registers!

Wanneer een gebruikersprogramma gestart wordt met de GO-toets, worden door het monitorprogramma de inhoud van de geheugenplaatsen GA, GX en GY gekopieerd in respectievelijk de accu, het X- en het Y-register. Pas daarna wordt het gebruikersprogramma gestart.
Normaal zal men een gebruikersprogramma altijd zelf initialiseren. Deze functie is echter van belang, als men bepaalde delen van het programma wil testen, waarbij dan de inhoud van de accu en de registers een vorig resultaat voorstellen. Dit "vorig" resultaat kan men dan ingeven in de genoemde geheugenplaatsen (ØØAD, ØØAE en ØØAF).

Retroforum

Mini-cursus MPS 65 computersysteem

Computers werken in een binair (= tweetallig) getalstelsel. Hierbij is het grondtal 2.
Bij het normale decimale stelsel is het grondtal tien. Een getal is opgebouwd uit een aantal cijfers, die afhankelijk van hun plaats aangeven hoe vaak een bepaalde macht van 10 in het totale getal voorkomt.

Dus 1984 wil zeggen:

$4 \times 10^0 = 4 \times 1$
$8 \times 10^1 = 8 \times 10$
$9 \times 10^2 = 9 \times 100$ en
$1 \times 10^3 = 1 \times 1000$

Omdat het grondtal 10 is, zijn er ook 10 cijfers (0 t/m 9) om aan te geven hoe vaak een bepaalde macht voorkomt.

In het binaire stelsel zijn er slechts twee cijfers (0 en 1) om aan te geven hoe vaak een bepaalde macht van 2 voorkomt.

De binaire getal 1011 wil zeggen:

$1 \times 2^0 = 1 \times 1 = 1$
$1 \times 2^1 = 1 \times 2 = 2$
$0 \times 2^2 = 0 \times 4 = 0$
$1 \times 2^3 = 1 \times 8 = 8$
+
11

Het binaire getal 1011 komt dus overeen met het decimale getal 11.

Voor computers is het binaire stelsel uitermate geschikt, daar de twee getallen 0 en 1 op eenvoudige wijze in elektrische toestanden kan worden omgezet (aan/uit; 0 Volt/+5 V; stroom/geen stroom; e.d.).

Voor de programmeur echter is het binaire stelsel wat minder eenvoudig omdat het erg lange getallen oplevert, die bovendien, omdat er alleen nullen en enen in voorkomen, erg onoverzichtelijk zijn.
Bijvoorbeeld: decimaal 1984 is binair 11111000000.

Elk cijfer in het binaire stelsel heet een bit. Door nu 4 bits te combineren, ontstaat een getal, dat nibble genoemd wordt. Een nibble kan $2^4 = 16$ verschillende waarden aannemen. Deze worden weergegeven door de cijfers 0 t/m 9 en de letters A t/m F.

Een getal voorgesteld door meerdere nibbles heeft als grondtal 16. We noemen zo'n getal hexadecimaal.

Dus 07C0 hexadecimaal is:

$0 \times 16^0 = 0 \times 1 = 0$
$C \times 16^1 = 12 \times 16 = 192$
$7 \times 16^2 = 7 \times 256 = 1792$
$0 \times 16^3 = 0 \times 4096 = 0$
+
1984 decimaal

Om aan te geven dat een getal hexadecimaal is wordt het vaak gevolgd door de letter H. Dus 07C0 H.

Voor de meeste computers bestaan assemblers, die de mnemonic-code van instructies en de daarbij behorende data en adressen omzetten in machinecode (hexadecimale/binaire getallen). Deze assemblers gebruiken vaak het $-teken om aan te geven dat een getal hexadexi-

6.1 Microcomputer MPS 65

Retroforum

maal is. Dit $-teken wordt dan voor het getal geplaatst.

Veel van de populaire microprocessoren werken met een databus, die 2 nibbles (= 8 bits) breed is. Twee nibbles worden een byte genoemd.
Voor de volledigheid zij nog vermeld, dat 2 bytes (= 16 bits) een woord wordt genoemd en 2 woorden (= 4 bytes = 32 bits) een dubbel woord of longword.

De volgende tabel verduidelijkt de samenhang tussen decimale, binaire en hexadeximale getallen.

decimaal	binair	hexadecimaal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Nu volgen een aantal voorbeelden van werken met hexadecimale en binaire getallen.

Voorbeeld 1:

AND-functie met 4 bits.

binair	hexadecimaal
0101	$5
∧0110	∧$6
0100	$4

Voorbeeld 2:

AND-functie met 8 bits.

binair	hexadecimaal
1101.1100	$DC
∧1100.0111	∧$C7
1100.0100	$C4

Voorbeeld 3:

ORA-functie (OR).

binair	hexadecimaal
0111.0000	$70
∨0011.0100	∨$34
0111.0100	$74

Voorbeeld 4:

EOR-functie (exclusieve OR).

binair	hexadecimaal
0101.0000	$50
⊻0011.0100	⊻$34
0110.0100	$64

Voorbeeld 5:

ADD-functie (optellen)

binair	hexadecimaal
0101.0010	$52
+0101.1100	+$5C
1010.1110	$AE

Voorbeeld 6:

ADC-functie (optellen met "carry").

binair	hexadecimaal
0101.0010	$ 52
+ 1101.0110	+ $ D6
1.0010.1000	$1.28

↑ Carry bit (1 onthouden)

Retroforum

Het carry bit geeft een overdracht aan naar het negende bit. D.w.z. voor overdracht naar een eventuele optelling van een meer significant deel van de op te tellen getallen.

N.B. De 6502 microprocessor heeft een mode, waarbij decimaal wordt gerekend. In de binaire mode werkt de carry-functie als in voorbeeld 6. D.w.z. dat het carry bit wordt "geset" (logisch "1" wordt) als de waarde $FF wordt overschreden. Bij decimaal rekenen gebeurt dit echter als de waarde $99 overschreden wordt. De mode verandert met de instructies SED (set Decimal) en CLD (clear Decimal). Het is zaak bij een gebruikersprogramma zeker te zijn van de mode waarin men werkt. Het is dan ook een goede gewoonte om een programma te beginnen met CLD en die delen van het programma, waarin bewust decimaal gerekend moet worden te beginnen met SED en te beëindigen met CLD.

CPU-registers.

De CPU (central processing unit) 6502 heeft een aantal registers, waarin bewerkingen kunnen worden uitgevoerd. Een van de registers is de accumulator, kortweg accu genoemd. Deze accumulator heeft een breedte van 8 bits (1 byte), daarnaast is er nog een carry-bit, die door de accu vaak als negende bit bij bewerkingen wordt gebruikt.

Om bewerkingen op twee getallen uit te voeren, moet er altijd één van de twee in de accu aanwezig (geladen) zijn. Het resultaat van de bewerking staat na de bewerking in de accu.

Voorbeeld 3 als programma.

Om de ORA-functie van voorbeeld 3 uit te voeren moeten de volgende acties ondernomen worden:

1. Laad de accu met het hexadecimale getal 70.
2. Voer een OR-functie uit tussen het getal in de accu en een getal op een geheugenplaats.
3. Het resultaat staat in de accu.

Deze beschrijving voldoet wel voor de programmeur, maar niet voor de computer. Deze heeft binaire code nodig. Door gebruik te maken van mnemonics (afkortingen) is het mogelijk het programma compact te schrijven, daarna kan het eenvoudig worden gecodeerd in binaire (of hexadecimale) code (handmatig of met behulp van een assembleerprogramma). De fabrikant heeft voor de CPU mnemonics en de daarbij behorende codes vastgelegd.
In mnemonic-vorm zou het programma uit voorbeeld 3 er als volgt kunnen uitzien:

```
LDA  #$70
ORA  #$34
BRK
```

Hierbij zijn LDA, ORA en BRK instructies, eventueel gevolgd door data.

\# is een assembler-code, die immediate adressing aangeeft, d.w.z. de te bewerken data staat op de geheugenplaats die volgt op de geheugenplaats van de instructie zelf.

$ is het teken dat aangeeft dat het hier om een hexadecimaal getal gaat.

Zoeken we in de 6502 instructietabel de bijbehorende codes op, dan kunnen we

6.1 Microcomputer MPS 65

Retroforum

van het programma de volgende "listing" maken:

adres	hex-code	instructie	operand
0200	A9	LDA	#$70
0201	70		
0202	09	ORA	#$34
0203	34		
0204	00	BRK	

Als dit programma in de MPS 65 wordt ingevoerd en daarna op adres 0200 gestart, worden de drie instructies afgewerkt.
De BRK-instructie zorgt voor een terugkeer naar het monitorprogramma, waarbij de monitor de inhoud van X-register, Y-register en de accu op respectievelijk displays 1 en 2, 3 en 4, 5 en 6 laat zien.

Displays:

(X-Reg.)	(Y-Reg.)	(Accu)
0 0	0 0	7 4

Voorbeeld 5 als programma

Om zeker te zijn van het juiste resultaat bij voorbeeld 5, moet voorafgaand aan de optelling in het programma het carry-bit nul gemaakt worden, er is immers nog geen sprake van een overdracht van een vorig resultaat.

Met behulp van de 6502 instructietabel schrijven we het programma en de erbij behorende code:

adres	hex-code	instructie	operand
0200	A9	LDA	#$52
0201	52		
0202	18	CLC	
0203	69	ADC	#$5C
0204	5C		
0205	00	BRK	

Na invoering van dit programma staat het resultaat van de optelling in de displays 5 en 6 (accu-inhoud).

In de tot nu toe gebruikte voorbeelden stonden de getallen die als factoren in de bewerking werden gebruikt in het programma zelf (onmiddellijk na de instructie).
Deze vorm van adresseren van de factoren ((operands in computertaal) noemt men immediate adressering.
Echter de 6502 CPU kent een veelheid van adresseringsmogelijkheden (zie ook tabel 4).

In het voorafgaande voorbeeld is nog een tweede adresseringsvorm aanwezig, de zogenaamde impliciete adressering, namelijk de CLC-instructie (clear carry = maak het carry-bit 0). De instructie impliceert, waarop deze betrekking heeft. Deze instructies bestaan dan ook slechts uit één byte. Andere voorbeelden zijn CLD, TAX, PHA, PLA etc.

Een andere adresseermogelijkheid is die, waarbij de instructie wordt gevolgd door het adres van de geheugenplaats, waar de data zich bevindt. Deze wijze van adresseren staat bekend als absolute adressering. De instructie wordt gevolgd door twee bytes, het eerste daarvan bevat het minst significante byte van het adres, het daaropvolgende bevat het meest significante byte.

De opdracht: laad de accu met de inhoud van geheugenplaats 0300 wordt in mnemonic LDA $0300 en in hex-code:

AD	Instructie
00	adresbyte (laag)
03	adresbyte (hoog)

6.1 Microcomputer MPS 65

Retroforum

Voorbeeld 7:

De inhoud van geheugenplaatsen 0300 en 0301 moet 'ge-OR-ed" worden. Het resultaat van deze OR-functie moet worden opgeborgen in geheugenplaats 0302. Het programma zelf begint op geheugenplaats #0200.

adres	hex-code	instructie	operand
0200	AD	LDA	$0300
0201	00	16 bits adres	
0202	03		
0203	2D	AND	$0301
0204	01	16 bits adres	
0205	03		
0206	8D	STA	$0302
0207	02	16 bits adres	
0208	03		
0209	00	BRK	

Als we in ons voorbeeld met de OR-functie de volgende getallen willen bewerken:

```
   0110.1001      $69
   0011.1100      $3C
∧              ∧
   0010.1000      $28
```

dan moeten we met behulp van de ADDRES-toets en de ENTER-toets de waarden $69 en $3C invoeren in respectievelijk de geheugenplaatsen $0300 en $0301.

Wederom met behulp van de ADDRES-toets en vervolgens de GO-toets wordt het programma op adres $0200 gestart. Na afloop zal de waarde $28 in geheugenplaats $0302 staan.

ZERO-Page adressering

De eerste 256 bytes van het geheugen (de adressen ØØØØ-ØØFF) worden pagina nul of zero-page genoemd.
Adressen op pagina nul kunnen natuurlijk via de eerder besproken absolute adressering worden bereikt.
Echter de 6502 CPU kent een speciale instructieset, voor instructies, die aan pagina 0 refereren. Het voordeel van deze instructies is, dat zij slechts twee bytes geheugenruimte innemen en dat zij sneller uitgevoerd worden.
De instructie LDA #$0010
(in hex-code AD 1000)
kan ook worden geschreven als zero-page-instructie
LDA $10 (in hex-code A 5 # 10)

Voorbeeld 8:

Als voorbeeld zeven, echter in plaats van de adressen 0300, 0301 en 0302 worden nu de adressen 0010, 0011 en 0012 gebruikt.

adres	hex-code	instructie	operand adres
0200	A5	LDA	$10
0201	10	(8 bit adres)	
0202	25	AND	$11
0203	11	(8 bit adres)	
0204	85	STA	$12
0205	12	(8 bit adres)	
0206	00	BRK	

Geïndexeerde adressering.

Bij geïndexeerde adressering wordt de inhoud van X- of Y-register mede gebruikt bij de bepaling van het operand adres.

Retroforum

Niet alle instructies hebben zowel X- als Y-indexering.

Voorbeeld 9:

Kopieer de inhoud van geheugenplaatsen Ø2ØØ-Ø2ØØ naar Ø3ØØ-Ø3ØD

adres	hex-code	instructie
0200	A2	LDX #$00
0201	00	
0202	BD	LDA $0200, X
0203	00	
0204	02	
0205	9D	STA $0300, X
0206	00	
0207	03	
0208	E8	INX
0209	E0	CPX #0E
020A	0E	
020B	D0	BNE $0202
020C	F5	
020D	00	BRK

Na de instructie LDX $00 is de inhoud van het X-register 00.
De instructies LDA $0200 en STA $0300 hebben betrekking op de adressen 0200 respectievelijk 0300.
Na uitvoering van de instructie INX is de inhoud van het X-register 01 geworden. De CPX #$0E-instructie vergelijkt de inhoud van het X-register met de waarde $0E. Aangezien dit ongelijk is zal de daaropvolgende BNE-instructie de computer terugvoeren naar adres 0202. De LDA- en STA-instructies worden opnieuw uitgevoerd, echter nu op adressen 0201 respectievelijk 0301 (0200 + X-register = 0201 en 0300 + X-register = 0301). Dit gaat door, totdat het X-register de waarde $0E bereikt, dan wordt niet meer teruggesprongen naar adres 0202 aangezien de CPX-instructie in een "equal" resulteert (Zero-flag wordt "1"). De laatste gekopieerde geheugenplaats is dan 020D (0200 + X-register = 0200 + 0D = 020D).

Ook de geïndexeerde adressering kent zero-page-instructies. Bij het bestuderen van de 6502 instructie-tabel valt op, dat er hier slechts twee Y-indexeringsinstructies zijn, namelijk die instructies die op het X-register zelf werken en waardoor dus uiteraard het X-register zelf niet gebruikt kan worden, als indexeringsregister.

De indexed X indirect adressering.

Eerst indexeren, dan indirect. Deze adressering is een vorm van zero-page adressering.
De instructiecode wordt gevolgd door een adres op pagina nul. Bij dit adres op pagina nul wordt de inhoud van het X-register opgeteld. De data die op het op deze wijze ontstane adres staat, vormt samen met de data van de volgende geheugenplaats het absolute adres waar de operand staat. Dit klinkt ingewikkeld, het volgende voorbeeld dient ter verduidelijking.

Voorbeeld 10:

adres	hex-code	instructie + operand (adres)
0200	A2	LDX 02
0201	02	
0202	A1	LDA ($20, X)
0203	20	
0204	00	BRK
0205	EA	NOP
0020	--	
0021	--	
0022	05	
0023	02	

6.1 Microcomputer MPS 65

Retroforum

Bij uitvoering van dit programma zal de computer de volgende stappen ondernemen bij de instructie LDA.

Bij adres 0020 wordt de inhoud van het X-register opgeteld, dit levert op $0020 + $02 = $0022. Nu wordt gekeken welk adres op geheugenplaatsen 0022 en 0023 is opgeslagen. Dit blijkt adres 0205 te zijn. Nu wordt de data die in geheugenplaats $0205 staat in de accu geladen. Dit is dus $EA.

De indirect, Y indexed adressering.

Eerst indirect, dan indexeren.

Aan de hand van het voorbeeld zal ook deze adresseringsvorm duidelijk worden.

Voorbeeld 11:

adres	hex-code	instructie + operand (adres)
0200	A0	LDY $05
0201	05	
0202	B1	LDA ($20), Y
0203	20	
0204	00	BRK
0205	EA	NOP
0020	00	
0021	02	

Bij dit voorbeeld voert de computer de volgende handelingen uit bij de instructie LDA.

Er wordt gekeken welk 16 bits adres op de geheugenplaatsen 0020 en 0021 staat. Dit blijkt te zijn 0200. Hierbij wordt de inhoud van het X-register opgeteld. Dit levert op $0200 + $05 = $0205. Nu wordt de inhoud van geheugenplaats 0205 in de accu geladen. Dit is dus $EA.

Retroforum

Aansluitingen van de uitbreidingsconnector J1

De uitbreidingsconnector bevat de aansluitingen van de uitgebreide SMP-bus.

Aan-sluit-punt Rij A	signaal	aan-sluit-punt Rij C	signaal
1	− 15V	1	− 12V
2	− 5V	2	GND
3	s.o.	3	+ 5V
4	Ø2	4	—
5	R/W	5	A12
6	RES	6	AØ
7	SYNC	7	A13
8	R/W	8	A1
9	—	9	A14
10	RAM R/W	10	A2
11	—	11	A15
12	READY	12	A3
13	BUSEN	13	—
14	DØ	14	A4
15	—	15	—
16	D1	16	A5
17	—	17	—
18	D2	18	A6
19	—	19	—
20	D3	20	—
21	D3	21	A7
22	—	22	—
23	D4	23	A8
24	IRQ	24	—
25	D5	25	A9
26	—	26	—
27	D6	27	A10
28	—	28	—
29	D7	29	A11
30	—	30	—
31	—	31	—
32	+ 15V	32	GND
	+ 5V		+ 12V

Tabel 4/6.1 -6:
Uitgebreide SMP-Bus aansluitingen.

De standaard adres- en data-lijnen, de stuurbussignalen en de voedingsspanningen zijn hierbij op gestandaardiseerde punten aangesloten, zodat uitbreidingskaarten, die dezelfde busstructuur hebben eenvoudig via een moederboard of een bedrade groep connectoren met de MPS 65 worden verbonden. De busconnector is een 64-polige DIN connector. Rij b is niet gebruikt. Zie tabel 6.
Een streepje betekent, dat de betreffende pen niet is aangesloten.

6.1 Microcomputer MPS 65

Retroforum

Het monitorprogramma van de MPS 65.

```
;### ZERO-PAGE ###

;*******************
;* ANWENDER-BEREICH*
;*******************
*=$00A0
ADR16 *=*+2
DAT8  *=*+1

*=$00A3
DISP1 *=*+1
DISP2 *=*+1
DISP3 *=*+1
DISP4 *=*+1
DISP5 *=*+1
DISP6 *=*+1

*=$00A9
STD   *=*+1
MIN   *=*+1
SEK   *=*+1
T004  *=*+1

*=$00AD
GA    *=*+1
GX    *=*+1
GY    *=*+1

*=$00C0
RESF  *=*+1
RES65 *=*+2

*=$00C3
IRQF  *=*+1
IRQ65 *=*+2

*=$00C6
NMIF  *=*+1
NMI65 *=*+2

*=$00C9
IRQMF *=*+1
IRQM  *=*+2

*=$00CC
BRKUF *=*+1
BRKU  *=*+2

*=$00FD
PAXY  *=*+3 ;P/AXY


;*******************
;*  SYSTEMBEREICH  *
;*******************
*=$0080
REC

*=$0100
BUP
*=$00D3
KALT  *=*+2
UHRR  *=*+2

*=*+12 ;RECORDER
;
KEY   *=*+1
FGPB  *=*+1
FGHEX *=*+1
FGENT *=*+1

R0    *=*+1
R1    *=*+1
R2    *=*+1
R3    *=*+1

RKX   *=*+1
RK1   *=*+1
RI1   *=*+1
;
;## END ZERO-PAGE ##
```

```
B     =$F600
ARBRAM=$0200
ZEIT1 =2501-2
;
V     =$A000
PORTB =V
PORTA =V+1
DIRPB =V+2
DIRPA =V+3
LATC1L=V+4
LATC1H=V+5
TIM1L =V+6
TIM1H =V+7
TIM2L =V+8
TIM2H =V+9
SR    =V+$A
ACR   =V+$B
PCR   =V+$C
IFR   =V+$D
IER   =V+$E

TG=$51
TS=$52
TI=$62
TL=$54
TB=$64
TA=$58
TE=$68

*=$0200
ANFMON
;-SYSTEMEINSPRUENGE-
JMP KEYIN+B
JMP KEYIN1+B
JMP DISPEX+B
JMP DISPSY+B
JMP PHAXY+B
JMP PLAXY+B
JMP ASK2+B
JMP ASK4+B
JMP MON65+B
JMP MON65E+B
JMP T015+B
JMP INC16X+B
JMP DEC16X+B
JMP ERR+B
RTS
.BYT 0,0
JMP STRICH+B
JMP SUCHEX+B

;===================
.BYT $26
.BYT '(C)BY THALER'
.BYT '10/1983'
;===================

;#### PROGRAMME ####

ERR ;ERROR-(A) (0-F)
  JSR STRICH+B
  AND #$0F
  TAY
  LDX #5
  LDA DISTAB+B,Y
ER1
  STA DISP1,X
  LDA TER+B-1,X
  DEX
  BPL ER1
  JSR KEYINE+B
  JSR DISPSY+B
  RTS
;
;
INDEXB ;ABS.=>REL.AD
  LDX #ADR16;BRANCH?
  JSR DEC16X+B
  LDX #0
  LDA (ADR16,X)
  LDX #ADR16
  JSR INC16X+B
  LDX #7
BTAB1
  CMP BTAB+B,X
  BEQ BOK
  DEX
  BPL BTAB1
  LDA #4 ;ERROR4
  JSR ERR+B
  JMP WARM1+B
```

```
BOK
  LDA ADR16 ;SICHERN
  PHA
  LDA ADR16+1
  PHA

  JSR STRICH+B
  LDA #$2F ;B
  STA DISP5
  LDA #9 ;R
  STA DISP6
  JSR ASK4+B

  PLA
  STA R2
  PLA
  STA R1

  LDA ADR16;REL.SPR.
  SEC
  SBC R1
  STA R3;LH
  LDA ADR16+1
  SBC R2 ;HB

  BMI BNEG
  BNE BERR2
  DEC R3 ;POS.
  LDA R3
  CMP #128
  BCS BERR2;>=
  LDX #0
  STA (R1,X);REL.ADR
  JMP BEND+B

BNEG
  CMP #$FF
  BNE BERR3
  DEC R3
  LDA R3
  BPL BERR3
  LDX #0
  STA (R1,X)

BEND
  LDA R1
  STA ADR16
  LDA R2
  STA ADR16+1
  JMP WARM1+B
BERR2
  LDA #3
  BNE BERR4
BERR3
  LDA #2
BERR4
  JSR ERR+B
  JMP BEND+B
BTAB
.BYT $90,$B0,$F0,$30
,$D0,$10,$50,$70

INDEXC ;SHIFT
  LDX #$FF
IC1
  DEX
  LDA $200,X
  STA $201,X
  LDA #$EA
  STA $200,X
  CPX ADR16
  BNE IC1
  INC ADR16
  JMP WARM1+B
;
PHAXY
  CLC
  BCC PLAXY+1
PLAXY
  SEC
  STA PAXY+2
  PLA
  STA PAXY
  PLA
  STA PAXY+1
  BCS P1
  LDA PAXY+2
  PHA
  TXA
  PHA
  TYA
```

```
  PHA
  LDA PAXY+2
  BCC P2
P1
  PLA
  TAY
  PLA
  TAX
  PLA
P2
  INC PAXY
  BNE *+4
  INC PAXY+1
  JMP (PAXY)
;
ASCII ;(A)=>(A),(X)
  PHA
  JSR ASC1+B
  TAX
  PLA
  ROR A
  ROR A
  ROR A
  ROR A
  JSR ASC1+B
  RTS
ASC1
  AND #$0F
  CMP #$0A
  BCS ASC2
  ORA #$30
  RTS
ASC2
  ADC #$36
  RTS
;
;##### ZEITEN ######
T015 ;(A)*0,15
  JSR PHAXY+B
  LDY #75
  LDX #255
  DEX
  BNE *-1
  DEY
  BNE T015+5
  SEC
  SBC #1
  BNE T015+3
  JSR PLAXY+B
   RTS
;
;*******************
;*TASTATUR, ANZEIGE*
;*INITIALISIERUNGEN*
;*******************

MON65E ;NACH ENTER
  JSR KEYINE+B

MON65
  LDX #$FF
  TXS
  JSR INIT1+B ;VIA
  LDA #$33
  CMP KALT
  BNE INITK
  CMP KALT+1
  BEQ WARMST
INITK
  STA KALT
  STA KALT+1
  JSR INIT2+B
  JSR INIT4+B

;MPS-65
  JSR DISPEX+B
  LDX #5
D65
  LDA TAB65+B,X
  STA DISP1,X
  DEX
  BPL D65

  JSR KEYINE+B ;ENT?

  JSR DISPSY+B
```

6.1 Microcomputer MPS 65

Retroforum

```
;SELBSTSTART?
  DEC $E003 ;01?
  BNE WARMST
  INC $E004 ;FF?
  BNE WARMST
  JSR STRICH+B
  JMP $E005

WARMST
  LDA RESF
  BNE *+5
  JMP (RES65)
  JSR DISPSY+B
  LDA #<ARBRAM
  STA ADR16
  LDA #>ARBRAM
  STA ADR16+1
WARM1
  LDX #0
  LDA (ADR16,X)
  STA DAT8
TASTEN
  JSR KEYIN+B
  CMP #TA
  BEQ ADRTST
  CMP #TE
  BEQ ENTTST
  CMP #TB
  BNE *+5
  JMP BSTTST+B
  CMP #TI
  BNE *+5
  JMP INDEX+B
  CMP #TG
  BEQ GOTST
  BNE DATTST
;
ADRTST
  LDA #0
  STA FGENT
  JSR STRICH+B
  LDA #$7D;A
  STA DISP5
  LDA #$1F;D
  STA DISP6
  JSR ASK4+B
  JMP ENT1+B
;
ENTTST
  LDA DAT8
  LDX #0
  STA (ADR16,X)
  LDX #ADR16
  JSR INC16X+B
ENT1
  LDX #0
  LDA (ADR16,X)
  STA DAT8
  JMP TASTEN+B
;
GOTST
  LDA ADR16
  STA R1
  LDA ADR16+1
  STA R2
  LDA #0;0 FUER ANWE
  STA ADR16
  STA ADR16+1
  STA DAT8
  JSR STRICH+B
  LDA GA
  LDX GX
  LDY GY
  JMP (R1)
;
BSTTST
  LDX #ADR16
  JSR DEC16X+B
  JMP ENT1+B
;
DATTST
  LDX #$FF
  STX FGENT
DAT1
  INX
  CMP TTAB+B,X
  BNE DAT1
TSTGEF
  TXA
  ASL DAT8
  ASL DAT8
  ASL DAT8
  ASL DAT8
  ORA DAT8
  STA DAT8
  JMP TASTEN+B
```

```
;
INDEX ;VERTEILER
  JSR STRICH+B
  LDA #$14;1
  STA DISP5
  LDA #$0D;N
  STA DISP6
  JSR KEYIN+B
  JSR DISPSY+B
  CMP #$41;F
  BNE *+5
  JMP UHR+B
  CMP #$21;D
  BNE *+5
  JMP UHRA+B
  CMP #$42;B
  BNE *+5
  JMP INDEXB+B
  CMP #$31;E
  BNE *+8
  JSR STRICH+B
  JMP $E008 ;IC5
  CMP #$11;C
  BNE *+5
  JMP INDEXC+B;SHIFT
  JMP MON65+B
;
KEYINE
  JSR KEYIN+B
  CMP #TE
  BNE KEYINE
  RTS
KEYIN
  LDA KEY
  BEQ KEYIN
KEYIN1
  LDA KEY
  PHA
  LDA #0
  STA KEY
  PLA
  RTS
;
;********************
;*   IRQ MONITOR    *
;********************

IRQMON
  PHA
  TXA
  PHA
  TYA
  PHA

  TSX
  LDA $104,X
  AND #$10
  BEQ IRQ3 ;NO BRK
  JMP IRQBRK+B

IRQ3
  BIT IFR
  BVS IRQ4 ;NO IRQ
  LDA IRQF ;IRQ65?
  BNE IRQ4
  JMP (IRQ65)

IRQ4
  LDA IRQMF ;IRQ+MON
  BNE IRQ1 ;NO IRQM
  JSR IRQ5+B
  JMP IRQ1+B ;RETURN

IRQ5
  JMP (IRQM)

IRQ1
;******
;********************
;* KEYBOARD'DISPLAY*
;********************
;
  LDA LATC1L
  CLI

  LDA FGHEX
  BNE NOHEX
  JSR DISP16+B
  JSR DISP8+B
NOHEX; NEUE SP.
  JSR REAPA+B
  CLC
```

```
  ADC #$10
  CMP #$70
  BCS RESS;GR/GL
  STA PORTA
  BCC SETB ;SET PB
;
RESS; SET SP.1
  AND #$90
  STA PORTA
;
SETB
  LDX FGPB
  BNE SCAN ;OUTPB

  AND #$70
  LSR A   ;=>
  LSR A
  LSR A
  LSR A
  TAX
  LDA DISP1-1,X
  STA PORTB
;
SCAN ;ABFRAGE TASTEN
  JSR REAPA+B
  LDX RK1 ;ABSPRUNG
  DEX
  BEQ ABFR1
  BPL ABFR2
;
ABFR0 ;STAENDIG.ABFR
  TAX
  AND #$0F
  BEQ INTE;KEINE TA
  STX RKX
  INC RK1
  BNE INTE

ABFR1 ;TASTE TESTEN
  TAX
  AND #$70
  STA RI1
  LDA RKX
  AND #$70
  CMP RI1
  BNE INTE;SP.FALSC
  TXA
  CMP RKX
  BNE ABFRR ;FEHLER
  STA KEY
  INC RK1
  JMP INTE+B

ABFR2 ;TASTE LOS?
  CMP RKX
  BEQ INTE;DERUECKT
  AND #$70
  STA RI1
  LDA RKX
  AND #$70
  CMP RI1
  BNE INTE;FALS.SPA

ABFRR ;ABFRAGE RESET
  LDA #0
  STA RK1

INTE
;
;******
; UHRWERK
  SED
  DEC UHRR ;1/400
  BNE UEN2
  DEC UHRR+1
  BPL UEN2
  LDA #<400
  STA UHRR
  LDA #>400
  STA UHRR+1
  LDY #0
  LDX #2
UHR1
  LDA STD,X
  CLC
  ADC #1
  CPX #0
  BNE UHR2
  CMP #$24
  BCS UHR3
  BCC UEN1
```

```
UHR2
  CMP #$60
  BCC UEN1
UHR3
  STY STD,X
  DEX
  BPL UHR1
  BMI UEN2
UEN1
  STA STD,X
UEN2
  CLD
;---

ENDINT
  PLA
  TAY
  PLA
  TAX
  PLA
  RTI
REAPA ;READ PORTA
  LDA #$7F
  AND PORTA
  EOR #$0F
  RTS
;####################
DISPEX
  PHA
  LDA #$FF
  BNE *+5
DISPSY
  PHA
  LDA #0
  STA FGHEX
  JSR INIT1+B
  PLA
  RTS
;
;********************
;* ANZEIGE ADRESSE *
;********************
DISP16
  LDA ADR16+1
  JSR SEGM+B
  STX DISP1
  STY DISP2
  LDA ADR16
  JSR SEGM+B
  STX DISP3
  STY DISP4
  RTS
;
;********************
;*  ANZEIGE DATUM   *
;********************
DISP8
  LDA DAT8
  JSR SEGM+B
  STX DISP5
  STY DISP6
  RTS
;
SEGM       ;ACCU
  TAY
  LSR A  ;=>
  LSR A
  LSR A
  LSR A
  TAX
  LDA DISTAB+B,X
  TAX
  TYA
  AND #$0F
  TAY
  LDA DISTAB+B,Y
  TAY      ;4-7 0-3
  RTS      ; X   Y

;********************
;*  SUCHE HEXZAHL   *
;*  DER TASTATUR    *
;********************
SUCHEX
  JSR KEYIN+B
  CMP #TB
  BEQ SUB
  LDX #0
```

6.1 Microcomputer MPS 65

Retroforum

```
SUC1
   CMP TTAB+B,X
   BEQ SUC2
   INX
   CPX #16
   BEQ SUCHEX
   BNE SUC1
SUC2
   TXA
   AND #$0F
   CLC
   RTS
SUB
   SEC
   RTS
;
STRICH
   JSR PHAXY+B
   JSR INIT3+B
   JSR DISPEX+B
   LDX #05
   LDA #$08;-
STR1
   STA DISP1,X
   DEX
   BPL STR1
   JSR PLAXY+B
   RTS
;
UHR
   JSR STRICH+B
   LDA #$37;U
   STA DISP5
   LDA #$2D;H
   STA DISP6
   JSR ASK4+B
   LDA ADR16+1
   STA STD
   LDA ADR16
   STA MIN
   LDA #0
   STA UHRR
   STA UHRR+1
   STA SEK
UHRA
   JSR DISPSY+B
UHR0
   LDA STD
   STA ADR16+1
   LDA MIN
   STA ADR16
   LDA SEK
   STA DAT8
   JSR KEYIN1+B
   CMP #TE
   BNE UHR0
   JMP WARMST+B
;
IRQBRK ;00=BREAK
   LDA BRKUF
   BNE *+5
   JMP (BRKU)

   JSR DISPSY+B
   TSX
   LDA $103,X
   STA DAT8
   LDA $101,X
   STA ADR16
   LDA $102,X
   STA ADR16+1
   JMP MON65E+B
;
ASK4;LIST/SCHR. 4ANZ.
   JSR SUCHEX+B
   BCS *-3
   ASL A
   ASL A
   ASL A
   ASL A
   STA R1
   LDA DISTAB+B,X
   STA DISP1
   LDA ADR16+1
   AND #$0F
   ORA R1
   STA ADR16+1 ;ANZ 1
AS1
   JSR SUCHEX+B
   BCC *+8
   LDA #8
   STA DISP1
   BNE ASK4
```

```
   STA R1
   LDA DISTAB+B,X
   STA DISP2
   LDA ADR16+1
   AND #$F0
   ORA R1
   STA ADR16+1 ;ANZ 2
ASK2
   JSR SUCHEX+B
   BCC *+8
   LDA #8
   STA DISP2
   BNE AS1
   ASL A
   ASL A
   ASL A
   ASL A
   STA R1
   LDA DISTAB+B,X
   STA DISP3
   LDA ADR16
   AND #$0F
   ORA R1
   STA ADR16 ;ANZ 3
   JSR SUCHEX+B
   BCC *+8
   LDA #8
   STA DISP3
   BNE ASK2
   STA R1
   LDA DISTAB+B,X
   STA DISP4
   LDA ADR16
   AND #$F0
   ORA R1
   STA ADR16 ;ANZ 4
   JSR DISPSY+B
   RTS
;
INC16X
   INC 0,X
   BNE *+4
   INC 1,X
   RTS

DEC16X
   PHA
   DEC 0,X
   LDA 0,X
   CMP #$FF
   BNE *+4
   DEC 1,X
   PLA
   RTS
;
INIT1 ;VIA
   LDX #$7F
   STX IER ;CLEAR IFR

   LDX #VTABE-VTABA-1
INIT11
   LDA VTABA+B,X
   STA V,X
   DEX
   BPL INIT11

INIT3
   LDX #0
   STX FGPB ;DISP.EIN
   CLD
   CLI
   RTS

ERRINT      ;ERROR 6
   JSR INIT2+B
   LDA #6
   ??MP ERR1+B
;
;*******************
;*  INIT VEKTOREN  *
;*******************

INIT2
   LDX #VEKE-VEKA-1
INIT21
   LDA VEKA+B,X
   STA RESF,X
   STA IRQF,X
   STA NMIF,X
```

```
   STA IRQMF,X
   STA BRKUF,X
   DEX
   BPL INIT21
   RTS

INIT4
   LDA #0
   STA GA
   STA GX
   STA GY
   LDX #9
CLEA1
   STA KEY,X
   DEX
   BPL CLEA1

   TAX ;RAM <=0
CLEA2
   STA $200,X
   STA $300,X
   STA $400,X
   STA $500,X
   STA $600,X
   STA $700,X
   INX
   BNE CLEA2
   RTS

NMI
   JSR PHAXY+B
   LDA NMIF
   BNE *+5
   JMP (NMI65)
   JMP ($FFFC)
;*******************
;*  INIT.-TABELLEN *
;*******************

VEKA
 .BYT $FF       ;FLAG
 .WOR ERRINT+B ;ERROR
VEKE

TAUF
 .BYT $7D,$37,$69,$0D

VTABA
 .BYT 0,$10,$7F,$70
 .BYT <ZEIT1,>ZEIT1
 .BYT 0,0,0,0,0,$40
 .BYT 0,0,$C0
VTABE

;*******************
;* TASTEN-DISP.CODE*
;*******************

TER ;'ERROR'
 .BYT $6B,9,9,$F,9
SPACE

*=ANFMON+$7D4
DISTAB;ANZ.0-F
 .BYT $77,$14,$5B,$5E
 ,$3C,$6E,$6F,$54,$7F
 ,$7E,$7D,$2F
 .BYT $63,$1F,$6B,$69
TAB65; MPS-65
 .BYT $39,$79,$6E,$08
 .BYT $6F,$6E
TTAB;0-F
 .BYT $18,$28,$38,$48
 ,$14,$24,$34,$44,$12
 ,$22,$32,$42
 .BYT $11,$21,$31,$41
;
;*******************
;-INTERRUPTVEKTOREN-
;*******************

*=ANFMON+$7FA
 .WOR NMI+B,MON65+B
 .WOR IRQMON+B

    END
```

Retroforum

de 65 C02 serie.

De hardware.

Er is tegenwoordig een C-mos versie van de 6502 microprocessor in de handel. Het meest voor de hand liggende voordeel is natuurlijk het geringere stroomverbruik. Van deze C-mos 6502 zijn versies beschikbaar voor vier verschillende klokfrequenties te weten 1, 2, 3 en 4 MHz maximaal. Het opgenomen vermogen bedraagt respectievelijk 20mW, 40mW, 60mW en 80mW, dat is zelfs bij de snelste versie een flinke verbetering tegenover de bipolaire 6502, die 575 mW dissipeerde. De C-mos versie kent een standby-mode, waarin hij slechts 10μW! dissipeert.
De in- en uitgangen van de C-mos 6502 blijven echter TTL-compatibel. Inherent aan C-mos is de grotere storingsongevoeligheid en de ruimere tolerantie op de voedingsspanning (5V ± 20% in plaats van 5V ± 10%).

Er zijn twee versies (65 C102 en C112) beschikbaar voor een multiprocessor omgeving. Deze hebben twee extra aansluitingen nl. BE (bus enable) en ML (memory lock). Het bus enable signaal zet de processor in tri-state (ontkoppeld van de bus) indien het laag gemaakt wordt. Hierdoor komt het geheugen vrij voor gebruik door andere processoren of bijvoorbeeld DMA (direct memory access).
De 6502 mag echter nooit toegang tot het memory worden ontzegd gedurende read-modify-write instructies. Tijdens de uitvoering van dergelijke instructies is de ML-uitgang laag. Door dit signaal is gebruiken in een "bus-controle-circuit", kan worden voorkomen, dat BE laag wordt als ML laag is.
De res (reset)-ingang van de C-mos versies is voorzien van een Schmitt-trigger, waardoor met behulp van een RC-netwerk er een eenvoudige "power-on: reset" is te realiseren.

De software.

De instructieset van de 65Cxx versie is uitgebreider dan die van zijn bipolaire broer. Er zijn een aantal nieuwe instructies en voor een aantal bestaande instructies is er een adresseervorm bijgekomen. Zie tabel 7.

1. De nieuwe adresseervormen

– Voor de instructies ADC, AND, CMP, EOR, LDA, ORA, SBC en STA is er nu een indirecte adressering via pagina nul mogelijk. D.w.z.: direct na de instructie in het programma staat een byte, dat een adres op pagina nul weergeeft, op dit adres en het daaropvolgende in pagina nul staat het adres van de operand.

Voorbeeld:

instructie	adres	code
LDA ($10)	0210	B2
	0211	10
NOP	0212	EA
	0010	12
	0011	02

Bij uitvoering van de LDA ($10) instructie kijkt de processor op adres 10 van pagina nul, daar vindt hij het lage adresbyte van de operand, vervolgens vindt hij het hoge adresbyte op adres 0011. De accu wordt nu geladen met de inhoud van het gevonden adres. Het gevonden adres is 0212, dus de accu-inhoud wordt EA.

Retroforum

– Voor de jump-instructie is er ook een nieuwe adresseervorm nl. de geïndexeerde indirecte sprong JMP (IND), X. In dit geval wordt de jump-instructie gevolgd door een indirect adres, waarop het sprongadres staat. Bij dit sprongadres wordt eerst nog de inhoud van het X-register opgeteld, waarna naar het adres wordt gesprongen.

Voorbeeld:

instructie	adres	code
LDX #$02	0200	A2
	0201	02
JMP (0300), X	0202	7C
	0203	00
	0204	03
	0300	--
	0301	--
	0302	12
	0303	04
	0412	--
	programma wordt van hieraf voortgezet.	

N.B.: in de C-mos versie is ook de fout verdwenen die in de bipolaire versie voorkwam bij de JMP (IND)-instructie. Als het lage adresbyte nu op locatie FF van een pagina staat, wordt het hoge adresbyte gelezen van adres 00 van de volgende pagina (bij de bipolaire 6502 werd dit gelezen van adres 00 van dezelfde pagina!).

2. De nieuwe instructies.

Een aantal nieuwe instructies geven de 65C02 meer mogelijkheden tot het testen van de status en het manipuleren van individuele bits.

BIT
Deze instructie is eigenlijk niet nieuw, maar heeft er een aantal adresseringsmogelijkheden bij. Behalve absoluut en zero-page kan de instructie nu ook met immediate, zero-page X-indexed en absolute X-indexed adressering worden gebruikt.

TRB
Test and reset memory bits with accu.
functie: A∧M naar M; M7 naar N; M6 naar V.
(A = accu, M = memory, N = negative flag, V = overflow flag)
adressering: ABS, Zero-page.

TSB
Test and set memory bits with accu.
functie: A∨M naar M; M7 naar N; M6 naar V.
adressering: ABS, Zero-page.

RMB0, RMB1 . . . . . RMB7
Reset memory bit (0, 1 . . . . . 7)
functie: maak memory bit 0.
adressering: alleen Zero-page.

SMBO, SMB1 . . . . . SMB7
Set memory bit (0, 1 . . . . . 7)
functie: maak memory bit 1.
adressering: alleen Zero-page.

BBR0, BBR1 . . . . . BBR7
Branch on memory bit reset (0, 1 . . . 7)
functie: test bit in geheugenplaats en branch als dit bit 0 is.
adressering: Zero-page voor gerefereerde geheugenplaats, relatief voor de sprong.
instructievorm: instructie.
laag adresbyte van pagina nul
sprongoffset.

6.1 Microcomputer MPS 65

Retroforum

BBS0, BBS1 . . . . . . BBS7
Branch on memory bit set (0, 1 . . . . . 7)
functie: test bit in geheugenplaats en branch als dit bit nul is.
adressering: als BBR.
instructievorm: als BBR.

BRA
Branch always.
functie: branch instructie zonder condities.
Deze instructie is nuttig bij het realiseren van relatieve (is plaats-onafhankelijke) programma's.

PHX – push (bewaar) X-register op de stack.

PHY – push (bewaar) Y-register op de stack.

PLX – pull x-register van stack. (haal een waarde van de stack en plaats deze in het X-register)

PLY – pull Y-register van stack, (haal een waarde van de stack en plaats deze in het Y-register)

STZ – store zero on memory.
functie: maak geheugenplaats 00.
adressering: absoluut; Zero-page; Zero-page X-indexed; absoluut X-indexed.

DEA – decrement accu. (verlaag accumulator met 1)

INA – increment accu. (verhoog accumulator met 1)

In tabel 7a en 7b zijn alle instructies met de erbij behorende codes vermeld.

In tabel 8 kan worden teruggevonden hoeveel klokcycli de processor voor een bepaalde instructie nodig heeft, zodat tijdkritische programma's van te voren op hun tijdsduur kunnen worden bekeken.

Alle niet gedefinieerde instructies worden door de C-mos versie van de 6502 behandeld als NOP. Bij de bipolaire versie is het gedrag niet altijd voorspelbaar.

6.1 Microcomputer MPS 65

Retroforum

	IMM	ABS	Z	ABS, X	ABS, Y	Z, X	Z, Y	(IND), Y	(IND), Y	(Z) IND	NB
ADC	2	4	3	4	4	4		6	5	5	(1)(2)
AND	2	4	3	4	4	4		6	5	5	(1)
ASL		6	5	7		6					
BIT	2	4	3	4		4					(1)
CMP	2	4	3	4	4	4		6	5	5	
CPX	2	4	3								
CPY	2	4	3								
DEC		6	5	7		6					
EOR	2	4	3	4	4	4		6	5	5	(1)
INC		6	5	7		6					
JMP		3						6		5	
JSR		6									
LDA	2	4	3	4	4	4		6	5	5	(1)
LDX	2	4	3		4		4				(1)
LDY	2	4	3	4		4					(1)
LSR		6	5	7		6					
ORA	2	4	3	4	4	4		6	5	5	
RMBn			5								
ROL		6	5	7		6					
ROR		6	5	7		6					
SBC	2	4	3	4	4	4		6	5	5	(2)
SMBn			5								
STA		4	3	5	5	4		6	6	5	
STX		4	3				4				
STY		4	3			4					
STZ		4	3	5		4					
TRB		6	5								
TSB		6	5								

IMPLIED INSTR.		BRANCH INSTR.	
BRK	7	BBRn	5
PHA	3	BBSn	5
PHP	3	BCC	2
PHX	3	BCS	2
PHY	3	BEQ	2
PLA	4	BMI	2
PLP	4	BNE	2
PLX	4	BPL	2
PLY	4	BRA	2
RTI	6	BVC	2
RTS	6	BVS	2

Alle andere impl. instr. vergen 2 klokpulsen.

NOTA BENE

1 Bij indexering 1 klokpuls meer indien paginagrens wordt overschreden

2 Bij decimaal rekenen 1 klokpuls meer.

Bij alle branch instructies.
1 klokpuls bijtellen als daadwerkelijk een sprong plaatsvindt en nogmaals 1 klokpuls extra indien daarbij een paginagrens wordt overschreden.

Tabel 4/6.1 -8 "Speed"tabel van de 6502/65C02

6.1 Microcomputer MPS 65

Retroforum

Memory referenced **instructies**

	IMM	ABS	Z	ABS, X	ABS, Y	Z, X	Z, Y	(IND), X	(IND), Y	(Z) IND
ADC	69	6D	65	7D	79	75		61	71	72
AND	29	2D	25	3D	39	35		21	31	32
ASL		0E	06	1E		16				
BIT	89	2C	24	3C		34				
CMP	C9	CD	C5	DD	D9	D5		C1	D1	D2
CPX	E0	EC	E4							
CPY	C0	CC	C4							
DEC		CE	C6	DE		D6				
EOR	49	4D	45	5D	59	55		41	51	52
INC		EE	E6	FE		F6				
JMP		4C						7C		6C
JSR		20								
LDA	A9	AD	A5	BD	B9	B5		A1	B1	B2
LDX	A2	AE	A6		BE		B6			
LDY	A0	AC	A4	BC		B4				
LSR		4E	46	5E		56				
ORA	09	ØD	05	1D	19	15		01	11	12
RMB0			07							
RMB1			17							
RMB2			27							
RMB3			37							
RMB4			47							
RMB5			57							
RMB6			67							
RMB7			77							
ROL		2E	26	3E		36				
ROR		6E	66	7E		76				
SBC	E9	ED	E5	FD	F9	F5		E1	F1	F2
SMBO			87							
SMB1			97							
SMB2			A7							
SMB3			B7							
SMB4			C7							
SMB5			D7							
SMB6			E7							
SMB7			F7							
STA		8D	85	9D	99	95		81	91	92
STX		8E	86				96			
STY		8C	84			94				
STZ		9C	64	9E		74				
TRB		1C	14							
TSB		0C	04							

Tabel 4/6.1 -7a: De instructies van de 65C02

6.1 Microcomputer MPS 65

Retroforum

Implied instructies	
ASL(A)	0A
BRK	00
CLC	18
CLD	D8
CLI	58
CLV	B8
DEA	3A
DEX	CA
DEY	88
INA	1A
INX	E8
INY	C8
LSR (A)	4A
NOP	EA
PHA	48
PHP	08
PHX	FA
PHY	5A
PLA	68
PLP	28
PLX	FA
PLY	7A
ROL(A)	2A
ROR(A)	6A
RTI	40
RTS	60
SEC	38
SED	F8
SEI	78
TAX	AA
TAY	A8
TSX	BA
TXA	8A
TXS	9A
TYA	98

Branching instructies		
Inst.	code	vlag status (F) memory page 0 bit status (M)
BBR0	0F	M0 = 0
BBR1	1F	M1 = 0
BBR2	2F	M2 = 0
BBR3	3F	M3 = 0
BBR4	4F	M4 = 0
BBR5	5F	M5 = 0
BBR6	6F	M6 = 0
BBR7	7F	M7 = 0
BBS0	8F	M0 = 1
BBS1	9F	M0 = 1
BBS2	AF	M2 = 1
BBS3	BF	M3 = 1
BBS4	CF	M4 = 1
BBS5	DF	M5 = 1
BBS6	EF	M6 = 1
BBS7	FF	M7 = 1
BCC	90	F - C = 0
BCS	B0	F - C = 1
BEQ	F0	F - Z = 1
BMI	30	F - Z = 0
BNE	D0	F - N = 1
BPL	10	F - N = 0
BRA	80	geen conditie
BVC	50	F - V = 0
BVS	70	F - V = 1

Zie ook processor status register
BBR en BBS instructies zijn pagina nul instructies:
instr.; pag 0-adres; offset.

Tabel 4/6.1 -7b: De instructies van de 65C02 (vervolg)

6.1 Microcomputer MPS 65

Retroforum

00	BRK	1	40	RTI	1	80	BRA	2	C0	CPY IMM	2
01	ORA (IND, X)	2	41	EOR (IND, X)	2	81	STA (IND, X)	2	C1	CMP (IND, X)	2
02	–	–	42	–	–	82	–	–	C2	–	–
03	–	–	43	–	–	83	–	–	C3	–	–
04	TSB Z	2	44	–	–	84	STY Z	2	C4	CPY Z	2
05	ORA Z	2	45	EOR Z	2	85	STA Z	2	C5	CMP Z	2
06	ASL Z	2	46	LSR Z	2	86	STX Z	2	C6	DEC Z	2
07	RMB0 Z	2	47	RMB4 Z	2	87	SMB0 Z	2	C7	SMB4 Z	2
08	PHP	1	48	PHA	1	88	DEY	1	C8	INY	1
09	ORA IMM	2	49	EOR IMM	2	89	BIT IMM	2	C9	CMP IMM	2
0A	ASL (A)	1	4A	LSR (A)	1	8A	TXA	1	CA	DEX	1
0B	–	–	4B	–	–	8B	–	–	CB	–	–
0C	TSB ABS	3	4C	JMP ABS	3	8C	STY ABS	3	CC	CPY ABS	3
0D	ORA ABS	3	4D	EOR ABS	3	8D	STA ABS	3	CD	CMP ABS	3
OE	ASL ABS	3	4E	LSR ABS	3	8E	STX ABS	3	CE	DEC ABS	3
0F	BBR0 Z	3	4F	BBR4 Z	3	8F	BBS0 Z	3	CF	BBS4 Z	3
10	BPL	2	50	BVC	2	90	BCC	2	00	BNE	2
11	ORA (IND), Y	2	51	EOR (IND), Y	2	91	STA (IND), Y	2	D1	CMP (IND), Y	2
12	ORA (IND)	2	52	EOR (IND), Z	2	92	STA (IND)	2	D2	CMP (IND), Z	2
13	–	–	53	–	–	93	–	–	D3	–	–
14	TRB Z	2	54	–	–	94	STY Z,X	2	D4	–	–
15	ORA Z,X	2	55	EOR Z,X	2	95	STA Z,X	2	D5	CMP Z,X	2
16	ASL Z,X	2	56	LSR Z,X	2	96	STX Z,Y	2	D6	DEC Z,X	2
17	RMB1 Z	2	57	RMB5 Z	2	97	SMB1 Z	2	D7	SMB5 Z	2
18	CLC	1	58	CLI	1	98	TYA	1	D8	CLD	1
19	ORA ABS, Y	3	59	EOR ABS, Y	3	99	STA ABS, Y	3	D9	CMP ABS, Y	3
1A	INA	1	5A	PHY	1	9A	TXS	1	DA	PHX	1
1B	–	–	5B	–	–	9B	–	–	DB	–	–
1C	TRB ABS	3	5C	–	–	9C	STZ ABS	3	DC	–	–
1D	ORA ABS, X	3	5D	EOR ABS, X	3	9D	STA ABS, X	3	DD	CMP ABS, X	3
1E	ASL ABS, X	3	5E	LSR ABS, X	3	9E	STZ ABS, X	3	DE	DEC ABS, X	3
1F	BBR1 Z	3	5F	BBR5 Z	3	9F	BBS 1 Z	3	DF	BBS5 Z	3
20	JSR	3	60	RTS	1	A0	LDY IMM	2	E0	CPX IMM	2
21	AND (IND, X)	2	61	ADC (IND, X)	2	A1	LDA (IND, X)	2	E1	SBC (IND, X)	2
22	–	–	62	–	–	A2	LDX IMM	2	E2	–	–
23	–	–	63	–	–	A3	–	–	E3	–	–
24	BIT Z	2	64	STZ Z	2	A4	LDY Z	2	E4	CPX Z	2
25	AND Z	2	65	ADC Z	2	A5	LDA Z	2	E5	SBC Z	2
26	ROL Z	2	66	ROR Z	2	A6	LDX Z	2	E6	INC Z	2
27	RMB2 Z	2	67	RMB6 Z	2	A7	SMB2 Z	2	E7	SMB6 Z	2
28	PLP	1	68	PLA	1	A8	TAY	1	E8	INX	1
29	AND IMM	2	69	ADC IMM	2	A9	LDA IMM	2	E9	SBC IMM	2
2A	ROL A	1	6A	ROR A	1	AA	TAX	1	EA	NOP	1
2B	–	–	6B	–	–	AB	–	–	EB	–	–
2C	BIT ABS	3	6C	JMP IND	3	AC	LDY ABS	3	EC	CPX ABS	3
2D	AND ABS	3	6D	ADC ABS	3	AD	LDA ABS	3	ED	SBC ABS	3
2E	ROL ABS	3	6E	ROR ABS	3	AE	LDX ABS	3	EE	INC ABS	3
2F	BBR2 Z	3	6F	BBR6 Z	3	AF	BBS2 Z	3	EF	BBS6 Z	3
30	BMI	2	70	BVS	2	B0	BCS	2	F0	BEQ	2
31	AND (IND), Y	2	71	ADC (IND), Y	2	B1	LDA (IND), Y	2	F1	SBC (IND), Y	2
32	AND (IND) Z	2	72	ADC (IND) Z	2	B2	LDA (IND) Z	2	F2	SBC IND	2
33	–	–	73	–	–	B3	–	–	F3	–	–
34	BIT Z,X	2	74	STZ Z,X	2	B4	LDY Z,X	2	F4	–	–
35	AND Z,X	2	75	ADC Z,X	2	B5	LDA Z,X	2	F5	SBC Z,X	2
36	ROL Z,X	2	76	ROR Z,X	2	B6	LDX Z,Y	2	F6	INC Z,X	2
37	RMB3 Z	2	77	RMB7 Z	2	B7	SMB3 Z	2	F7	SMB7 Z	2
38	SEC	1	78	SEI	1	B8	CLV	1	F8	SED	1
39	AND ABS, Y	3	79	ADC ABS, Y	3	B9	LDA ABS, Y	3	F9	SBC ABS, Y	3
3A	DEA	1	7A	PLY	1	BA	TSX	3	FA	PLX	1
3B	–	–	7B	–	–	BB	–	–	FB	–	–
3C	BIT ABS, X	3	7C	JMP (IND, X)	3	BC	LDY ABS, X	3	FC	–	–
3D	AND ABS, X	3	7D	ADC ABS, X	3	BD	LDA ABS, X	3	FD	SBC ABS, X	3
3E	ROL ABS, X	3	7E	ROR ABS, X	3	BE	LDX ABS, Y	3	FE	INC ABS, X	3
3F	BBR3 Z	3	7F	BBR7 Z	3	BF	BBS3 Z	3	FF	BBS7 Z	3

Tabel 4/6.1 -9: Decodeertabel voor 6502/65C02 programma's. Achter elke code staat vermeld hoeveel bytes de instructie inneemt.

Retroforum